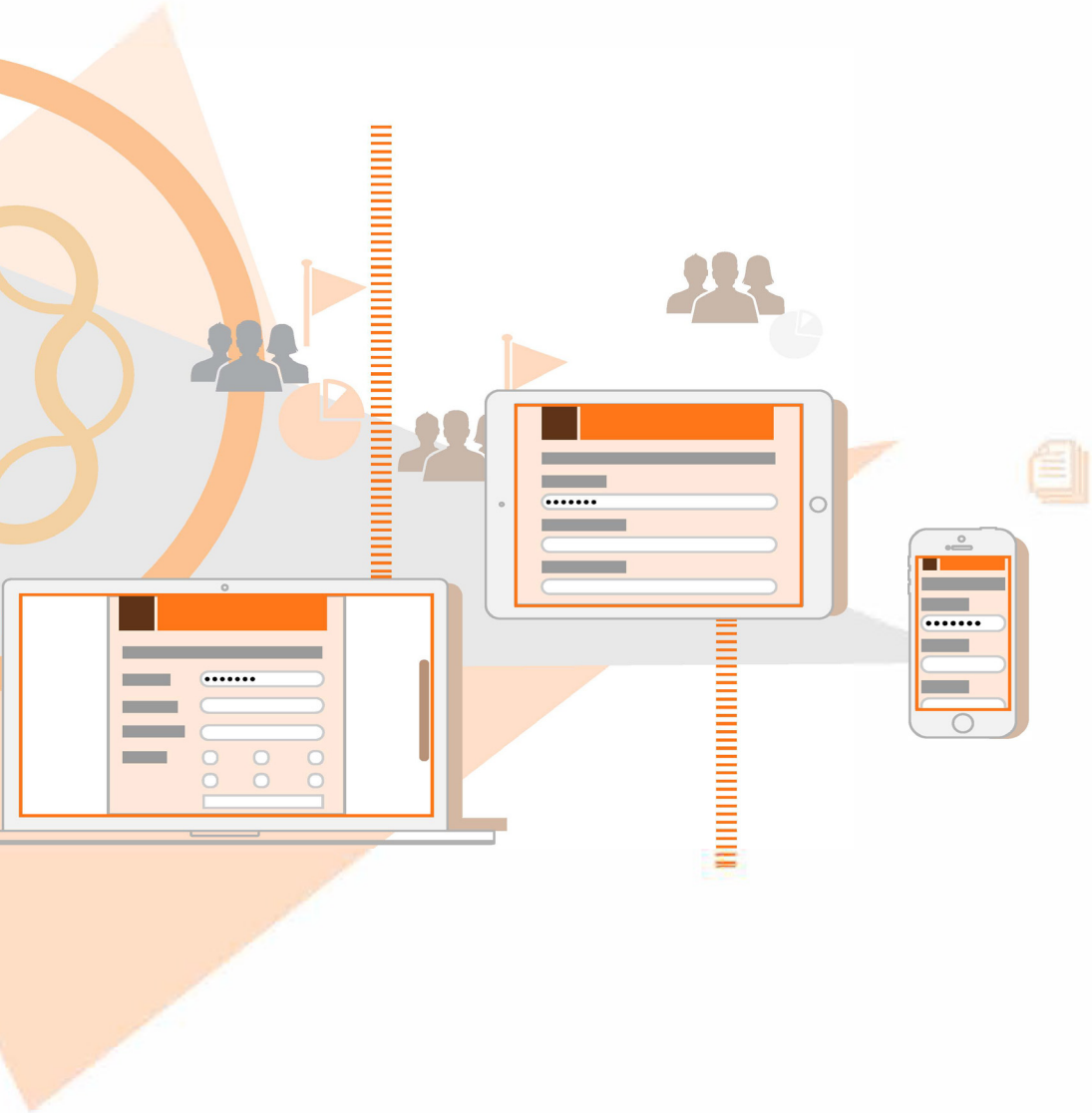
# AEM Forms Workbench Help



AEM 6.2 Forms

## Legal notices

For legal notices, see http://help.adobe.com/en_US/legalnotices/index.html.

# Contents


**Chapter: 1** **About AEM Forms Workbench . . . . . 1**

About the user interface . . . . . 1

Perspectives . . . . . 1

Views . . . . . 2

Editors . . . . . 2

Preferences . . . . . 2

Related software . . . . . 2

Adobe Community Help Client (CHC) . . . . . 3

**Chapter: 2** **What's new in Workbench . . . . . 4**

New for Workbench . . . . . 4

Process designer enhancements . . . . . 4

Looping in processes . . . . . 4

Grouping in processes . . . . . 4

Automating legacy solutions upgrade . . . . . 4

Archive Migration tool . . . . . 4

Upgrade legacy processes and artifacts . . . . . 4

Dependency detection . . . . . 5

Visualizing dependencies . . . . . 5

Asset renaming support . . . . . 5

Timer enhancements to Event Behavior Configuration . . . . . 5

Workbench offline notifications . . . . . 5

Performance enhancements . . . . . 5

Common variables . . . . . 5

Persisting XML form data with Data Models . . . . . 6

Integrating Data Models with Web Services . . . . . 6

Integrating Data Models with Data Services . . . . . 6

Building expressions with referred XSDs . . . . . 6

**Chapter: 3** **Leveraging legacy solutions in AEM forms . . . . . 7**

About the upgraded environment . . . . . 7

Review the AEM Forms run-time environment: . . . . . 8

Strategies for leveraging legacy solutions . . . . . 8
Continued execution . . . . . 8
Maintenance of legacy items . . . . . 8
To maintain legacy items: . . . . . 9
Progressive development . . . . . 9
To create AEM Forms assets from legacy items: . . . . . 9
Importing resources, processes, and events to applications . . . . . 9
Importing multiple versions of a process . . . . . .10
Importing from the run-time environment . . . . . .10
Import legacy items from the run-time environment: . . . . . 10
Importing from the file system . . . . . .11
Import from the file system: . . . . . 11
Exporting files referenced by literal values . . . . . .11
Importing from AEM Forms 8.x archive files . . . . . .12
Legacy LCA files that include service configurations . . . . . 12
Import legacy items from AEM Forms archive files: . . . . . 13
Maintaining legacy run-time instances . . . . . .13
Changing existing run-time instances . . . . . .13
Change legacy processes, events, and resources: . . . . . 13
Updating legacy solutions in other environments . . . . . .14
Selecting contents . . . . . 14
Create an archive file: . . . . . 14
Update an archive file: . . . . . 15
Upgrading legacy solutions to AEM forms . . . . . .16
Upgrading legacy artifacts . . . . . .16
Breaking the link to existing run-time instances . . . . . .17
Remove the value of Deployment ID: . . . . . 17
Creating endpoints for processes . . . . . .18
To add a start point: . . . . . 18
Replacing legacy subprocesses . . . . . .18
Replace legacy subprocesses: . . . . . 19
Configuring security settings . . . . . .19
Using deprecated operations . . . . . .19
Exporting embedded documents from operation properties . .19
Export embedded documents: . . . . . 19
Changing the reliance on null value results from XPath expressions 20
Upgrading human-centric processes . . . . . .20
Replacing form-specific variables . . . . . 20
Using legacy render and submit processes . . . . . 21
Configuring Workspace start points . . . . . 22
Using the User 2.0 service . . . . . 24
Migrating LCAs to AEM Forms . . . . . .27
Updating references to assets . . . . . .27
Process services . . . . . 27
Resources . . . . . 28

**Chapter: 4** **Using Workbench . . . . . 29**
Before you begin . . . . . .29
Process and form planning . . . . . .29
Assembling a development team . . . . . .29
Development process . . . . . .30
Logging In to a AEM Forms Server . . . . . .30
Logging in . . . . . .30
To log in to a server: . . . . . 31
Increasing the memory allocation . . . . . .31
Increase the allocated memory: . . . . . 31
Configuring server connections . . . . . .31
To configure a server: . . . . . 32
Logging out . . . . . .33
To log out of a server: . . . . . 33
User permissions . . . . . .33
Mutual authentication . . . . . .34
Working with Applications . . . . . .35
About the Applications view . . . . . .35
Developing applications in a multi-developer environment . . .36
Synchronize the local version with the version in the repository 36
Checking in applications or assets . . . . . 36
Checking out applications or assets . . . . . 37
Discarding the changes . . . . . 38
Viewing asset history . . . . . 38
Managing user access to applications . . . . . 38
Adding and removing applications . . . . . .39
Creating an application . . . . . 39
Adding applications from the server . . . . . 40
Removing applications . . . . . 40
Working with assets . . . . . .41
Adding and removing assets . . . . . 41
Organizing assets . . . . . 44
Editing and viewing assets . . . . . 45
Managing asset dependencies . . . . . 46
Applications and assets versioning . . . . . .47
Asset versions . . . . . 47
Deploying applications . . . . . .48
Deploy an application . . . . . 49
Redeploy an application: . . . . . 49
Undeploy an application: . . . . . 49
Moving applications into another environment . . . . . .50
About archive files . . . . . 50
Creating archive files . . . . . 50
Avoiding socket time-outs when creating archives . . . . . .53
Create archives asynchronously . . . . . 54
Creating Forms . . . . . .54

Create a form design using Workbench: . . . . . .55
Opening the Form Design perspective . . . . . .55
Open the Form Design perspective: . . . . . 55
Organizing your forms and assets . . . . . .56
Creating the form design . . . . . .56
Create a form design: . . . . . 57
New Form . . . . . 57
Specify Form Data Model . . . . . 57
Form Usage . . . . . 58
Opening Designer . . . . . 59
Opening form designs . . . . . .59
Open a form design: . . . . . 60
Close a form design: . . . . . 60
Saving the form design . . . . . .60
Where to find more information . . . . . .60
Creating XML Schemas . . . . . .61
Create an XML schema . . . . . 61
Edit an XML schema . . . . . 61
Managing Resources . . . . . .62
Filtering resources . . . . . .62
Filter the resources in the Resources view: . . . . . 62
Ensuring that the PDF icon is displayed . . . . . 63
Viewing resource relationships . . . . . .63
View the resource relationships: . . . . . 64
Sort the list: . . . . . 64
Working with file versions . . . . . .64
Viewing the version history . . . . . 64
Using older versions of files . . . . . 65
Setting access permissions . . . . . .65
About access permissions . . . . . 65
Viewing and changing access permissions . . . . . 66
Adding access permissions . . . . . 66
Adding different permissions to different subfolders . . . . . 67
Removing access permissions . . . . . 68
Managing Components and Services . . . . . .69
Opening the Components view . . . . . .69
Installing components . . . . . .69
To install a component: . . . . . 69
Patching components . . . . . .69
To patch a component: . . . . . 70
Starting components and services . . . . . .70
To start a component: . . . . . 70
To start a service: . . . . . 70
Activating services . . . . . .70
To activate a service: . . . . . 71
Editing service configurations . . . . . .71

To edit a service configuration: . . . . . 71
Removing service configurations . . . . . .71
To remove a service configuration: . . . . . 71
Stopping components and services . . . . . .71
To stop a component or service: . . . . . 71
Deactivating services . . . . . .72
To deactivate a service: . . . . . 72
Uninstalling components . . . . . .72
To uninstall a component: . . . . . 72
Customizing Perspectives . . . . . .72
Moving views . . . . . .72
Saving perspectives . . . . . .73
Restoring perspectives . . . . . .73

**Chapter: 5** **Getting started with process design . . . . . 74**
About Processes . . . . . .74
Opening the Process Design perspective . . . . . .74
To open the Process Design perspective: . . . . . 74
Process Design perspective views . . . . . .74
To open a view: . . . . . 75
Services . . . . . .75
Process diagrams . . . . . .76
Process data . . . . . .77
Process input and output data . . . . . .77
Process data model . . . . . .77
Access to process data . . . . . .78
Process design guidelines . . . . . .79
Order of implementation . . . . . .79
Process designs for reuse . . . . . .79
Reuse of variables . . . . . .80
Process execution . . . . . .80
Process instances . . . . . .80
Process modifications . . . . . 81
New processes . . . . . 81
Process life cycle . . . . . .81
Short-lived processes and long-lived processes . . . . . .81
Execution . . . . . 82
Client invocation . . . . . 82
Data persistence . . . . . 82
Branch types . . . . . 83
Transaction Propagation . . . . . .83
Process diagram modeling examples . . . . . .84
Sequential routing . . . . . .84
Parallel branches . . . . . .85
Synchronization . . . . . .85
Conditional routing . . . . . .86

Simple merge . . . . . .86
Multi-choice . . . . . .87
Gateway implementation . . . . . 87
Event implementation . . . . . 88
Synchronizing merge . . . . . .89
Implementation for a gateway multi-choice . . . . . 89
Implementation for an event multi-choice . . . . . 89
Multi-merge . . . . . .90
Loops . . . . . .91
Loop counters . . . . . 92
Process completion . . . . . .92
Multiple independent instances . . . . . .93
Gateway implementation . . . . . 93
Event implementation . . . . . 93
Multiple instances synchronization . . . . . .94
Gateway implementation . . . . . 94
Event implementation . . . . . 95

**Chapter: 6** **Process Quick Starts . . . . . 96**

Recommended skills . . . . . .96
Assigning tasks based on roles . . . . . .96
Prerequisites . . . . . .97
Configuration . . . . . .98
Other considerations . . . . . .99
Using form data with multiple forms . . . . . .99
Prerequisites . . . . . .99
Configuration when forms are similar . . . . . 100
Configuration when forms are not similar . . . . . 100
Other considerations . . . . . 101
Assembling multiple documents . . . . . 101
Prerequisites . . . . . 102
Configuration . . . . . 102
Retrieving the DDX File . . . . . 103
Assembling the document . . . . . 103
Other considerations . . . . . 104
Certifying policy-protected documents . . . . . 104
Prerequisites . . . . . 105
Configuration . . . . . 105
Other considerations . . . . . 106
Throwing events to initiate processes . . . . . 107
Prerequisites . . . . . 108
Configuration . . . . . 108
Other considerations . . . . . 109
Using Barcode Data in Processes . . . . . 110
Prerequisites . . . . . 112
Configuration . . . . . 113

Other considerations . . . . . 113
Sending output to a printer . . . . . 114
Configuration . . . . . 116
Input . . . . . 116
Batch Options . . . . . 116
Output . . . . . 116
Input . . . . . 116
Other considerations . . . . . 116
Creating pre-filled and interactive PDF forms . . . . . 117
Configuration . . . . . 118
Input . . . . . 118
PDF Options . . . . . 118
Output . . . . . 119
Input . . . . . 119
Other considerations . . . . . 119
Handling data submitted from a form . . . . . 119
Configuration . . . . . 121
Route Evaluation . . . . . 121
Input . . . . . 121
Form Submission Options . . . . . 121
Output . . . . . 122
Input . . . . . 122
Input . . . . . 122
Other considerations . . . . . 122
Applying usage rights to PDF documents using a watched folder . . . 123
Configuration . . . . . 124
Input . . . . . 124
Output . . . . . 124
General . . . . . 125
Server Configurations . . . . . 125
Inputs/Outputs . . . . . 125
Other considerations . . . . . 125

**Chapter: 7** **Creating and managing processes . . . . . .126**

Creating processes using the New Process wizard . . . . . 126
Create a process using the wizard: . . . . . 127
Configuring the Workspace start point . . . . . 128
Configure the Workspace start point: . . . . . 128
Configuring the Mobile start point (Deprecated) . . . . . 129
Configure the Mobile start point (Deprecated): . . . . . 129
Configuring the Email start point . . . . . 130
Configure the Email start point: . . . . . 130
Configuring the Watched Folder start point . . . . . 131
Configure the Watched Folder start point: . . . . . 131
Process editing aids . . . . . 131
Copying elements . . . . . 132

Copy an element: . . . . . 132
Changing the magnification of process diagrams . . . . . 132
Variable highlighting . . . . . 132
Adding annotations . . . . . 133
To add an annotation element: . . . . . 133
To modify the annotation element: . . . . . 133
To delete an annotation element: . . . . . 133
Creating processes based on existing process . . . . . 134
Create a process using copy and paste: . . . . . 134
Create a process using Save As: . . . . . 134
Managing processes . . . . . 135
Opening and closing processes . . . . . 135
Open the process: . . . . . 135
Close the process: . . . . . 135
Saving processes . . . . . 135
Save the process to local disk: . . . . . 136
Modifying processes . . . . . 136
Change process properties . . . . . 136
Change process properties: . . . . . 136
Deleting processes . . . . . 138
Delete a process: . . . . . 138
Activating and deactivating LiveCycle ES (8.x) processes . . . . . 138
Deactivate a process: . . . . . 139
Activate a process: . . . . . 139
Exporting and importing LiveCycle ES (8.x) processes . . . . . 139
Exporting processes . . . . . 139
Export a process: . . . . . 139
Importing processes . . . . . 139
Import a process: . . . . . 140
Grouping in Processes . . . . . 140
Grouping activities in your process . . . . . 140
Looping in Processes . . . . . 141
Adding a loop . . . . . 141
Understanding the loop . . . . . 141

**Chapter: 8** **Starting processes using start points . . . . . .143**

Add and configure a start point: . . . . . 144
Workspace start point properties . . . . . 144
General . . . . . 144
Presentation & Data . . . . . 145
Start Point Output . . . . . 146
Reader Submit . . . . . 146
Options . . . . . 146
Attachment Options . . . . . 147
Workspace User Interface . . . . . 147
Mobile start point (Deprecated) properties . . . . . 147

General . . . . . 148
Presentation & Data . . . . . 148
Start Point Output . . . . . 149
Options . . . . . 149
Email start point properties . . . . . 149
General . . . . . 149
Options . . . . . 150
Server Configurations . . . . . 151
Scheduling . . . . . 153
Inputs/Outputs . . . . . 153
Input . . . . . 153
Output . . . . . 154
Watched folder start point properties . . . . . 155
General . . . . . 155
Server Configurations . . . . . 156
Scheduling . . . . . 158
Inputs/Outputs . . . . . 159
Input . . . . . 159
Output . . . . . 160
Programmatic start point properties . . . . . 161

**Chapter: 9** **Process variables . . . . . .162**
Configuration parameters . . . . . 162
Creating variables . . . . . 163
To create a variable: . . . . . 164
Creating Common Variables . . . . . 165
Editing variables . . . . . 165
To edit a variable: . . . . . 166
Deleting variables . . . . . 166
To delete a process variable: . . . . . 166
Process data exposed to users . . . . . 166

**Chapter: 10** **Working with operations . . . . . .167**
Adding and deleting operations . . . . . 167
Add an operation: . . . . . 167
Delete an operation: . . . . . 168
Input and output data for operations . . . . . 168
Configuring input and output data . . . . . 169
Configure the input and output data for an operation: . . . . . 169
Data coercion . . . . . 169
Specifying template expressions . . . . . 171
Invoking subprocesses . . . . . 171
Add a subprocess: . . . . . 172
Abstract activities . . . . . 172
Adding, defining, and deleting abstract activities . . . . . 173

Add an abstract activity: . . . . . 174
Define an abstract activity: . . . . . 174
Delete an abstract activity: . . . . . 174
Organizing operations in swimlanes . . . . . 174
Adding, modifying, and deleting swimlanes . . . . . 175
Add a swimlane above or below the selected swimlane: . . . . . 175
Modify a swimlane: . . . . . 176
Delete a swimlane: . . . . . 176
Modifying pools . . . . . 176
Modify the name and description of the pool: . . . . . 176
Configuring the order of execution . . . . . 176
Specifying the start activity of a process . . . . . 177
To specify the start activity in a process diagram: . . . . . 177
Drawing routes to link operations . . . . . 177
Adding and deleting routes . . . . . 178
Modifying route shapes . . . . . 180
Modifying route labels . . . . . 180
Making decisions using routes . . . . . 181
Routing condition format . . . . . 181
Adding and modifying routing conditions . . . . . 181
Specifying the order of routes . . . . . 182
Adding branches using gateways . . . . . 183
Adding and deleting gateways . . . . . 183
To add a gateway element: . . . . . 184
To delete a gateway element: . . . . . 184
Branches . . . . . 185
Adding, editing, deleting, and copying branches . . . . . 185
To add a branch: . . . . . 185
To edit a branch: . . . . . 186
To copy a branch: . . . . . 186
To delete a branch: . . . . . 186
Adding elements to gateway branches . . . . . 186
To change which operation is executed first: . . . . . 187
Validating routes used by gateway branches . . . . . 187
Operation and branch compatibility . . . . . 187
Transactions . . . . . 187

**Chapter: 11** **Controlling process flow using events . . . . . .190**

Event types . . . . . 190
Event categories . . . . . 191
Throwing events . . . . . 192
Add and configure asynchronous event throws . . . . . 192
Add and configure timer event throws . . . . . 193
Receiving event throws . . . . . 194
Add and configure event receives . . . . . 194
Catching event throws . . . . . 195

Add event catches . . . . . 196
Handling event catches . . . . . 198
Handle event catches . . . . . 198
Event start points . . . . . 198
Add event start points . . . . . 199
Configure event start points . . . . . 199
Configure an event as a start point: . . . . . 199
Setting an event as the start activity . . . . . 200
Set events as the start activity . . . . . 200
Adding and defining abstract events . . . . . 200
Add abstract events . . . . . 200
Define abstract events . . . . . 200
Deleting events . . . . . 201
Delete events: . . . . . 201
Mapping data for events . . . . . 201
Adding event data to send . . . . . 201
Add event data and event message data . . . . . 202
Creating event filters . . . . . 202
Add event filters . . . . . 203
Edit an event filter . . . . . 203
Delete an event filter . . . . . 203
Storing event data to process variables . . . . . 204
Add a process data map . . . . . 204
Edit a process data map . . . . . 205
Delete a process data map . . . . . 205

**Chapter: 12 Preserving process results using the Archive Wizard . . . . . .206**

To start the Archive wizard: . . . . . 206
Archive wizard: Document and archival method properties . . . . . 207
Select a document to archive . . . . . 207
Conversion options . . . . . 208
How do you want to archive the document? . . . . . 208
Archive wizard: Email server properties . . . . . 209
Archive wizard: Email address properties . . . . . 210
Archive wizard: File system properties . . . . . 210
Archive wizard: Printer properties . . . . . 211
Archive wizard: Summary . . . . . 211

**Chapter: 13 Designing human-centric processes . . . . . .212**

Involving users in processes . . . . . 212
Enable users to start processes . . . . . 213
Send a task to one user . . . . . 213
Send tasks to multiple users simultaneously . . . . . 213
Document review and approval processes . . . . . 214
Workspace approval tools . . . . . 214

Collection data and XPath functions . . . . . 214
Completion policies . . . . . 215
Electronic signature processes . . . . . 215
Digital signatures . . . . . 215
Click-through electronic signatures . . . . . 215
Alternate tools for interacting with processes . . . . . 215
Email . . . . . 215
Mobile devices . . . . . 216
Designing data capture and presentation . . . . . 216
Assets for capturing and presenting data . . . . . 216
About captured data . . . . . 217
Data formats . . . . . 217
XDP form data . . . . . 218
Configuring XML variables for XDP data . . . . . 219
Saving XML as XDP data . . . . . 219
Displaying the xml data structure in XPath Builder . . . . . 220
About action profiles . . . . . 221
Default action profiles . . . . . 222
Modifying and creating action profiles . . . . . 222
Modify or create an action profile: . . . . . 222
Remove an action profile: . . . . . 223
Creating minimal processes for action profiles . . . . . 223
Action profile services at run time . . . . . 226
About prepare data services . . . . . 226
Advantages over subprocesses . . . . . 227
About render services . . . . . 227
Rendering to HTML and sending email . . . . . 228
About submit services . . . . . 228
Creating tasks . . . . . 229
Add an Assign Task or Assign Multiple Tasks operation: . . . . 229
Assigning tasks to users . . . . . 230
Reassigning tasks for out-of-office users . . . . . 231
Assign tasks for Assign Task operations: . . . . . 231
Assign tasks for Assign Multiple Tasks operations: . . . . . 231
Select a specific user . . . . . 232
Select a specific group . . . . . 232
Select a user list . . . . . 233
Use an XPath expression or variable to specify users . . . . . 233
Creating and changing user lists . . . . . 234
User list general properties . . . . . 234
User list population properties . . . . . 235
Configuring the presentation of task data . . . . . 236
Asset and data . . . . . 236
Document . . . . . 236
PDF assets with no submit buttons . . . . . 236
Use an asset and data: . . . . . 236

Use a document variable: . . . . . 237
Providing task instructions . . . . . 237
Provide task instructions: . . . . . 238
Providing actions for submitting tasks . . . . . 238
Add a user action . . . . . 239
Adding destinations for Assign Task operation actions . . . . . 239
Adding completion policies to Assign Multiple Tasks operations 240
Creating completion policies . . . . . 241
Saving task data . . . . . 242
Task Result Collection and the Workspace Approval Container (Deprecated) . . . . . 242
Save task data for Assign Task operations: . . . . . 243
Save task data for Assign Multiple Tasks operations: . . . . . 243
Configuring task functionality . . . . . 243
Make opening tasks optional . . . . . 244
Make opening tasks optional: . . . . . 244
Maximizing the form or Guide . . . . . 244
Maximize the form or Guide . . . . . 244
Require confirmation when submitting tasks . . . . . 245
Add a confirmation message to a user action: . . . . . 245
Specifying the Workspace user interface . . . . . 245
Specify Workspace tools: . . . . . 246
Setting time constraints . . . . . 246
About business calendars . . . . . 247
Sending reminders about tasks . . . . . 247
Setting deadlines for tasks . . . . . 248
Escalating tasks . . . . . 249
Overriding task notification settings . . . . . 250
Configure email notifications: . . . . . 250
Creating email templates . . . . . 251
Workspace URL parameters . . . . . 253
Enable task completion by replying to notification email . . . . . 254
Configuring task delegations and consultations . . . . . 254
Restrict delegation or consultation of tasks: . . . . . 255
Configuring the sharing of forwarded tasks . . . . . 255
Configuring the sharing of claimed tasks . . . . . 255
Modify the Share Tasks For Shared Queues process: . . . . . 256
Configuring access to task functionality . . . . . 257
Default ACL . . . . . 258
ACLs and shared queues . . . . . 258
ACLs and forwarded tasks . . . . . 258
Specifying users . . . . . 258
Add an ACL for a user: . . . . . 259
Extend the ACL to shared queues: . . . . . 259
Delete a user from the access control list: . . . . . 259
Configuring attachments and notes . . . . . 259

Configure attachments and notes: . . . . . 260
Specifying task priority . . . . . 260
Specify task priority: . . . . . 260
Working with captured data . . . . . 261
Assessing review and approval results . . . . . 261
Retrieving form data from XMl variables . . . . . 262
Using submitted data with Guides or Flex applications and forms 263
Document attributes for attachments and notes . . . . . 264
Adding data nodes to xml variables . . . . . 265
Best practices for Workspace . . . . . 266
Use short category names . . . . . 266
Limit the number of process cards per category . . . . . 267
Use effective card information . . . . . 268
Card titles . . . . . 268
Task instructions on task cards . . . . . 269
Image size in process cards . . . . . 270
Email notifications . . . . . 270
Submit button . . . . . 271
Creating approval processes using Approval wizard . . . . . 271
Create a process using the Approval wizard . . . . . 271
Best practices for mobile devices . . . . . 272
Using certified signatures in forms in Workspace . . . . . 273
Modifying the Render PDF Form process to use certified signatures in a dynamic form with Workspace . . . . . 273
To change the Render PDF Form process: . . . . . 274
Using certified forms with Acrobat or Adobe Reader version 6 or earlier 274

**Chapter: 14 Using data models with processes . . . . . .275**

Using data models for process input and output variables . . . . . 275
Using data models with form and Guide data . . . . . 275
Integrating Data Models with Web Services and Data Services . . . . . 276
Integrating with Web Services . . . . . 276
Integrating with WSDL/SOAP-based Web Service using a Data Model 276
Integrating with WSDL-based Web Service using WSDL Service Reference 276
Integrating with REST/HTTP-based Web Services . . . . . 276
Integrating with Data Services . . . . . 277
Integrating with an existing database . . . . . 277
Integrating with a new database . . . . . 277

**Chapter: 15 Creating XPath expressions . . . . . .279**

XPath Builder (Process Properties) . . . . . 279
XPath Builder (Data Mapping) . . . . . 281
Location . . . . . 281

Expression . . . . . 281
Building expressions by using XPath Builder . . . . . 282
Editing text in the expression work area . . . . . 282
To edit text in the expression work area: . . . . . 282
Building simple XPath expressions . . . . . 283
To build an expression that uses process variables: . . . . . 283
Building XPath expressions that use functions . . . . . 285
To build an expression that uses functions: . . . . . 285
Building XPath expressions that use operators . . . . . 288
To build an expression that uses operators: . . . . . 288
Building XPath expressions that use Common Variables . . . . 289
To build an expression that uses common variables: . . . . . 290
Examples of XPath expressions . . . . . 290
Retrieving form field values . . . . . 291
Changing values in form data . . . . . 292
Retrieving node sets . . . . . 293
Converting data types . . . . . 294
Converting XML to string data . . . . . 294
Converting to string data . . . . . 294
Converting string data to XML . . . . . 295
Accessing data in data collections . . . . . 295
list variables . . . . . 295
map variables . . . . . 296
Using XPath expressions as list indexes and map keys . . . . . 297
XPath function reference . . . . . 298
Special characters in function syntax . . . . . 298
Boolean functions . . . . . 298
boolean . . . . . 299
false . . . . . 299
not . . . . . 299
true . . . . . 300
Collection functions . . . . . 300
empty-list . . . . . 300
empty-map . . . . . 301
get-map-keys . . . . . 302
get-collection-size . . . . . 302
get-list-item-count . . . . . 302
get-list-item-percentage . . . . . 303
get-map-values . . . . . 304
get-map-values-for-keys . . . . . 304
Date-Time functions . . . . . 305
add-days-to-date . . . . . 305
add-days-to-dateTime . . . . . 305
add-hours-to-dateTime . . . . . 306
add-minutes-to-dateTime . . . . . 306
add-months-to-date . . . . . 307

add-months-to-dateTime . . . . . 307
add-seconds-to-dateTime . . . . . 308
add-years-to-date . . . . . 308
add-years-to-dateTime . . . . . 309
current-date . . . . . 309
current-dateTime . . . . . 310
current-time . . . . . 310
date-equal . . . . . 310
date-greater-than . . . . . 311
date-less-than . . . . . 311
dateTime-equal . . . . . 312
dateTime-greater-than . . . . . 312
dateTime-less-than . . . . . 313
get-day-from-date . . . . . 313
get-day-from-dateTime . . . . . 313
get-days-from-date-difference . . . . . 314
get-days-from-dateTime-difference . . . . . 314
get-hours-from-dateTime . . . . . 315
get-hours-from-dateTime-difference . . . . . 315
get-hours-from-time . . . . . 316
get-minutes-from-dateTime . . . . . 316
get-minutes-from-dateTime-difference . . . . . 316
get-minutes-from-time . . . . . 317
get-month-from-date . . . . . 317
get-month-from-dateTime . . . . . 318
get-months-from-date-difference . . . . . 318
get-months-from-dateTime-difference . . . . . 318
get-seconds-from-dateTime . . . . . 319
get-seconds-from-dateTime-difference . . . . . 319
get-seconds-from-time . . . . . 320
get-timezone-from-date . . . . . 320
get-timezone-from-dateTime . . . . . 321
get-timezone-from-time . . . . . 321
get-year-from-date . . . . . 321
get-year-from-dateTime . . . . . 322
get-years-from-dateTime-difference . . . . . 322
get-years-from-date-difference . . . . . 322
format-date . . . . . 323
format-dateTime . . . . . 324
parse-date . . . . . 324
parse-dateTime . . . . . 325
time-equal . . . . . 325
time-greater-than . . . . . 325
time-less-than . . . . . 326
Document functions . . . . . 326
getDocLength . . . . . 326

getDocAttribute . . . . . 327
setDocAttribute . . . . . 327
getDocContentType . . . . . 328
setDocContentType . . . . . 328
getDocContentBase64 . . . . . 329
getDocFromBase64 . . . . . 329
Miscellaneous . . . . . 330
deserialize . . . . . 330
is-null . . . . . 330
serialize . . . . . 331
Node set functions . . . . . 331
count . . . . . 331
last . . . . . 331
position . . . . . 332
sum . . . . . 332
Number functions . . . . . 332
ceiling . . . . . 332
floor . . . . . 333
number . . . . . 333
round . . . . . 333
String functions . . . . . 334
concat . . . . . 334
contains . . . . . 334
ends-with . . . . . 335
lower-case . . . . . 335
normalize-space . . . . . 335
starts-with . . . . . 335
string . . . . . 336
substring-after . . . . . 336
substring-before . . . . . 337
substring . . . . . 337
string-length . . . . . 338
translate . . . . . 338
upper-case . . . . . 339
Operators . . . . . 339
Date and time parameters . . . . . 340
Time zone descriptive identifiers . . . . . 341
A . . . . . 341
B . . . . . 351
C . . . . . 352
E . . . . . 352
G . . . . . 355
H . . . . . 355
I . . . . . 356
J . . . . . 356
K . . . . . 356

L . . . . . 356
M . . . . . 356
N . . . . . 357
P . . . . . 357
R . . . . . 358
S . . . . . 359
T . . . . . 359
U . . . . . 359
V . . . . . 360
W . . . . . 360
Z . . . . . 360

**Chapter: 16 Testing and troubleshooting process versions . . . . . .361**

Validation reports . . . . . 361
Reviewing validation messages . . . . . 362
To see documentation for a validation message: . . . . . 362
To change the way that validation messages are grouped: . . . . 363
To change which validation messages appear: . . . . . 363
To display the location of the problem: . . . . . 363
To copy validation messages: . . . . . 363
To refresh the validation messages for a process: . . . . . 363
Customizing validation behavior . . . . . 363
To customize validation behavior: . . . . . 364
Validation message reference . . . . . 364
DCI-007 . . . . . 364
DCI-008 . . . . . 365
DCI-009 . . . . . 365
DCI-010 . . . . . 366
DCI-011 . . . . . 366
DCI-012 . . . . . 367
DCI-013 . . . . . 367
WBP-001 . . . . . 368
WBP-003 . . . . . 368
WBP-007 . . . . . 369
WBP-008 . . . . . 369
WBP-009 . . . . . 369
WBP-010 . . . . . 370
WBP-013 . . . . . 370
WBP-018 . . . . . 371
WBP-019 . . . . . 371
WBP-020 . . . . . 371
WBP-033 . . . . . 371
WBP-034 . . . . . 372
WBP-035 . . . . . 372
WBP-036 . . . . . 373
WBP-037 . . . . . 373

WBP-038 . . . . . 373
WBP-039 . . . . . 373
WBP-040 . . . . . 374
WBP-101 . . . . . 374
WBP-102 . . . . . 375
WBP-104 . . . . . 375
WBP-105 . . . . . 375
WBP-107 . . . . . 376
WBP-109 . . . . . 376
WBP-110 . . . . . 377
WBP-111 . . . . . 377
WBP-112 . . . . . 378
WBP-120 . . . . . 378
WBP-150 . . . . . 378
WBP-151 . . . . . 379
WBP-152 . . . . . 379
WBP-153 . . . . . 380
WBP-170 . . . . . 380
WBP-177 . . . . . 380
WBP-178 . . . . . 381
WBP-179 . . . . . 381
WBP-180 . . . . . 381
WBP-181 . . . . . 382
WBP-182 . . . . . 382
WBP-183 . . . . . 382
WBP-184 . . . . . 382
WBV-001 . . . . . 383
WBV-002 . . . . . 383
Recording and playing back process . . . . . 383
Enabling and disabling recording . . . . . 384
To enable or disable recording for a process version: . . . . . 384
Invoking process versions . . . . . 385
Invoking processes directly . . . . . 385
Invoking processes using endpoints . . . . . 386
Playing back process recordings . . . . . 386
Opening recordings in playback mode . . . . . 386
Playing recordings . . . . . 388
Interpreting recordings . . . . . 389
Deleting recordings . . . . . 390
To delete a recording: . . . . . 390
Customizing record and playback behavior . . . . . 391
To customize record and playback behavior: . . . . . 392
Limiting storage space used for recordings . . . . . 392
Configuring server memory for large recordings . . . . . 393
Testing using realistic scenarios . . . . . 393
Test cases to consider . . . . . 394

Handling errors in the production environment . . . . . 395
Execution errors . . . . . 395
Stalled operation errors . . . . . 395
Stalled branch errors . . . . . 396
Implementation errors . . . . . 396
Computational errors . . . . . 396
Situational errors . . . . . 396
Monitoring variables using the Variable Logger service . . . . 396
Handling errors using exception event catches . . . . . 397
Handling situational errors using the Stall service . . . . . 398
Inspecting application server logs . . . . . 398
Edit server log viewer configuration settings . . . . . 398

**Chapter: 17** **Managing event types . . . . . .400**
Event type properties . . . . . 400
Display the properties of event types . . . . . 401
Custom event types . . . . . 401
Create custom event types . . . . . 402
Delete custom event types . . . . . 403
Creating data schemas for event data and event message data . . . . 403
Data schemas for event data . . . . . 404
Data schemas for event message data . . . . . 404
Organizing event types . . . . . 405
Displaying event types in categories or groups . . . . . 405
Exporting and importing event types . . . . . 405
Export event types . . . . . 406
Import event types . . . . . 406
Activating and deactivating event types . . . . . 406
Deactivate event types . . . . . 406
Activate event types . . . . . 407
Enable Rights Management event types . . . . . 407
Event type reference . . . . . 407
Asynchronous event types . . . . . 408
Asynchronous event type summary . . . . . 408
RMAdminEvent . . . . . 410
RMDocumentEvent . . . . . 411
RMErrorEvent . . . . . 412
RMPolicyEvent . . . . . 414
RMPolicySetEvent . . . . . 415
RMServerEvent . . . . . 416
RMUserEvent . . . . . 418
StartReviewStage . . . . . 419
TaskAttachmentAdded . . . . . 420
TaskAttachmentDeleted . . . . . 421
TaskAttachmentInfoUpdated . . . . . 422
TaskAttachmentUpdated . . . . . 424

TaskCancelled . . . . . 425
TaskClaimed . . . . . 426
TaskCompleted . . . . . 427
TaskCompletedWithData . . . . . 428
TaskConsulted . . . . . 429
TaskCreated . . . . . 430
TaskDeadlined . . . . . 431
TaskEscalated . . . . . 432
TaskFormDataSaved . . . . . 433
TaskForwarded . . . . . 434
TaskLocked . . . . . 435
TaskReassigned . . . . . 436
TaskRejected . . . . . 437
TaskReminderSent . . . . . 438
TaskShared . . . . . 439
TaskTerminated . . . . . 440
TaskUnlocked . . . . . 441
TaskVisibilityChange . . . . . 442
TerminateReview . . . . . 443
Exception event types . . . . . 444
Exception . . . . . 444
Timer event types . . . . . 445

**Chapter: 18 (Deprecated) Guides . . . . . .446**

**Chapter: 19 Using (Deprecated) Document Builder . . . . . .447**

Getting Started with (Deprecated) Document Builder . . . . . 447
Create a DDX document . . . . . 447
Open an existing DDX document in an application . . . . . 447
Open a DDX document on your computer . . . . . 447
Copy an existing DDX document into (Deprecated) Document Builder 447
Assembling PDF Documents . . . . . 448
Create a DDX document that builds a simple PDF document . 448
Create a DDX document that adds a dynamic watermark . . . 449
Create a DDX document that builds a PDF Portfolio . . . . . 450
Set PDF result and source properties . . . . . 453
PDF result properties you can set from the Result panel . . . . . 453
PDF source properties you can set from the PDF Source panel 455
Assembling XDP Documents . . . . . 458
Create a DDX document that builds a simple XDP document . 459
Create a DDX document that inserts a form fragment into an XDP document . . . . . 460
Set XDP result and source properties . . . . . 461
XDP result properties you can set from the Result tab . . . . . 462
XDP source properties you can set from the XDP Source panel 462

Set XDP Content source properties . . . . . 465
XDP Content attributes you can set from the XDP Content panel 465
Resolve image references in source . . . . . 467
Resolve all image references in a source XDP document . . . . . 467
Resolve specified image references in a source XDP document 468
Selectively resolve absolute or relative references . . . . . 468
Validating the DDX Document . . . . . 468
Validate the DDX . . . . . 469
Troubleshooting tips . . . . . 469
Previewing the Result from a DDX Document . . . . . 469
Set up your system for previewing XDP results . . . . . 469
On Windows systems . . . . . 469
Preview the DDX result . . . . . 470
Editing the XML for the DDX Document . . . . . 470
Edit the XML . . . . . 470

**Chapter: 20 Creating device profiles using XDC Editor . . . . . .472**

About XDC files . . . . . 472
Role of XDC files . . . . . 473
XDC file contents . . . . . 473
Using XDC files . . . . . 474
Using XDC files in Designer . . . . . 475
Selecting a paper type for a master page . . . . . 475
Selecting fonts applied to text . . . . . 476
Determining whether you need to modify the Designer.xdc file 476
Considerations for printing a form using the installed XDC files 476
Adding new paper types to the Designer.xdc file . . . . . 477
XDC files installed . . . . . 478
Template XDC files for generic print description languages . . . 478
Sample XDC files for use with specific printers . . . . . 480
XDC file storage . . . . . 481
Creating and modifying XDC files . . . . . 481
Before you begin . . . . . 481
Finding the XDC files installed with SAP NetWeaver Application Server 481
Copying sample and template XDC files to your local hard drive 481
Finding documentation for your printer . . . . . 482
Copying XDC files into your application . . . . . 482
Opening an XDC file . . . . . 483
To open a local XDC file . . . . . 483
To open an XDC file in an application . . . . . 483
Opening an XDC file in Eclipse . . . . . 483
Specifying a name for the XDC file . . . . . 484
Specifying media and trays . . . . . 484
Specifying media types . . . . . 484
Determining or specifying the media used by a form . . . . . 485

Remove a medium . . . . . 486
Modify a medium (PCL and PostScript printers only) . . . . . 486
Identifying your printer's input tray numbers . . . . . 486
Trays (PCL and PostScript printers only) . . . . . 488
Media and tray mapping (PCL and PostScript Printers only) . . . 489
(PostScript- and PCL-based XDC files) Adding or changing output trays 490
Specifying font descriptions . . . . . 490
Preparing to add fonts . . . . . 490
Add fonts . . . . . 491
Remove a font . . . . . 492
Font sequences for label printers . . . . . 492
Specifying printer capabilities . . . . . 493
Specifying the language supported by a printer . . . . . 493
Specifying RFID characteristics (for printers that support ZPL only) 494
Specifying sequences . . . . . 495
PDL sequences . . . . . 495
Predefined variables . . . . . 495
Use of variables in sequences . . . . . 497
Encoding PCL sequences . . . . . 498
Modifying sequences . . . . . 498
Deploying XDC files . . . . . 499
Deploying XDC files to Designer installations . . . . . 499
Making XDC files available to Output . . . . . 500
To copy an XDC file from your local file system to the AEM forms server: 500
Determining PCL font sequences . . . . . 500
About escape sequences that specify printer-resident fonts . . 500
Text-parsing method for fonts . . . . . 501
Character encoding . . . . . 501
Font characteristics . . . . . 502
Combining escape sequences . . . . . 502
Combined sequence that specifies the Courier font . . . . . 503
Combined sequence that specifies the Coronet font . . . . . 503

**Chapter: 21 Variable types reference . . . . . .505**

Service-defined data types . . . . . 505
Complex data types . . . . . 505
Common variable properties . . . . . 505
Implicit properties . . . . . 505
Name . . . . . 506
Title . . . . . 506
Description . . . . . 506
Type . . . . . 506
Subtype . . . . . 506
General . . . . . 506
Required . . . . . 506

Input . . . . . 506
Output . . . . . 506
Enduser UI items . . . . . 507
Searchable . . . . . 507
Visible In UI . . . . . 507
AcrobatVersion . . . . . 507
Data items . . . . . 507
AcrobatVersion-OutputService . . . . . 508
Data items . . . . . 508
Datatype specific settings . . . . . 509
Default value . . . . . 509
ApproverTO . . . . . 510
Data items . . . . . 510
additionalMetadata . . . . . 510
completedBy . . . . . 510
completedFromIP . . . . . 510
createdAt . . . . . 510
disposition . . . . . 511
endDate . . . . . 511
finalComments . . . . . 511
id . . . . . 511
reviewContext . . . . . 511
reviewStage . . . . . 511
startDate . . . . . 511
status . . . . . 511
umOid . . . . . 511
AssemblerOptionSpec . . . . . 512
Data items . . . . . 512
defaultStyle . . . . . 512
failOnError . . . . . 512
firstBatesNumber . . . . . 512
logLevel . . . . . 512
validateOnly . . . . . 513
Datatype specific settings . . . . . 513
Job Log Level . . . . . 513
Validate Only . . . . . 514
Fail On Error . . . . . 514
Default Style . . . . . 514
First Bates Number . . . . . 514
AssemblerResult . . . . . 514
Data items . . . . . 515
documents . . . . . 515
failedBlockNames . . . . . 515
jobLog . . . . . 515
lastBatesNumber . . . . . 515
multipleResultsBlocks . . . . . 515

numRequestedBlocks . . . . . 515
successfulBlockNames . . . . . 515
successfulDocumentNames . . . . . 516
throwables . . . . . 516
binary . . . . . 516
boolean . . . . . 516
Datatype specific settings . . . . . 516
byte . . . . . 517
Business Calendar Date . . . . . 517
Data items . . . . . 517
calendar . . . . . 517
date . . . . . 517
days . . . . . 517
endTime . . . . . 517
startDate . . . . . 518
Datatype specific settings . . . . . 518
Number of Business Days . . . . . 518
Business Calendar Name . . . . . 518
CentralResult . . . . . 518
Data items . . . . . 518
logDoc . . . . . 518
responseDoc . . . . . 519
resultDoc . . . . . 519
traceDoc . . . . . 519
Certificate Encryption Option Spec . . . . . 519
Data items . . . . . 519
compat . . . . . 520
option . . . . . 520
CertificateInformation . . . . . 520
Data items . . . . . 520
certificate . . . . . 521
revocationInformation . . . . . 521
trusted . . . . . 521
CertificatePath . . . . . 521
Data items . . . . . 521
certificateInformation . . . . . 521
failureReason . . . . . 521
status . . . . . 522
Character Set (barcoded forms) . . . . . 522
Character Set (File Utilities) . . . . . 523
ContentAccessPermision (deprecated) . . . . . 523
Data items . . . . . 524
authority . . . . . 524
authority Type . . . . . 524
isAllowed . . . . . 524
permission . . . . . 524

CRCResult (deprecated) . . . . . 524
Data items . . . . . 524
attributeMap . . . . . 524
browseLink . . . . . 525
contentMimeType . . . . . 525
creationDate . . . . . 525
creator . . . . . 525
document . . . . . 525
folderpath . . . . . 526
modificationDate . . . . . 526
modifier . . . . . 526
nodeName . . . . . 526
nodeType . . . . . 526
nodeUuid . . . . . 526
qualifiedNodePath . . . . . 526
storeName . . . . . 526
storeScheme . . . . . 526
title . . . . . 526
versionLabel . . . . . 526
CreateDocumentResultType . . . . . 526
Data items . . . . . 527
absoluteFileUrl . . . . . 527
relativeFileUrl . . . . . 527
fileUploadSuccessful . . . . . 527
Credential . . . . . 527
Data items . . . . . 527
alias . . . . . 527
certificate . . . . . 527
credentialType . . . . . 527
spiName . . . . . 528
Datatype specific settings . . . . . 528
Use SPI . . . . . 528
Alias . . . . . 528
SPI Name . . . . . 528
Certificate . . . . . 528
SPI Properties . . . . . 528
CRLOptionSpec . . . . . 529
Data items . . . . . 529
alwaysConsultLocalURL . . . . . 529
goOnline . . . . . 529
ignoreValidityDates . . . . . 529
LDAPServer . . . . . 529
localURI . . . . . 530
requireAKI . . . . . 530
revocationCheckStyle . . . . . 530
Datatype specific settings . . . . . 530

Consult Local URI First . . . . . 530
Local URI for CRL Lookup . . . . . 530
Revocation Check Style . . . . . 530
LDAP Server . . . . . 531
Go Online for CRL Retrieval . . . . . 531
Ignore Validity Dates . . . . . 531
Require AKI Extension in CRL . . . . . 531
CustomAttributeTO . . . . . 531
Data items . . . . . 531
attrKey . . . . . 531
attrValue . . . . . 532
id . . . . . 532
reviewContext . . . . . 532
reviewTemplate . . . . . 532
DataAndMetaDataDescriptor . . . . . 532
date . . . . . 532
literal value . . . . . 532
Example . . . . . 532
dateTime . . . . . 533
Literal value . . . . . 533
Example . . . . . 533
decimal . . . . . 533
Literal value . . . . . 533
Example . . . . . 534
Delimiter . . . . . 534
document . . . . . 534
Datatype Specific Settings . . . . . 534
DocInfo . . . . . 535
Data items . . . . . 535
attributeNameValue . . . . . 535
content . . . . . 535
objectType . . . . . 535
pid . . . . . 535
version . . . . . 535
Document Form . . . . . 535
Data items . . . . . 536
document . . . . . 536
formURL . . . . . 536
Datatype specific settings . . . . . 536
URL . . . . . 536
Advanced Settings . . . . . 536
Data items . . . . . 537
Enable Render Service . . . . . 537
Call The Render Service Only Once . . . . . 537
Service Name and Operation Name . . . . . 537
Service Input . . . . . 537

Service Output . . . . . 537
Submit Service . . . . . 537
DocumentTO . . . . . 538
double . . . . . 538
Literal value . . . . . 538
Encryption Identity . . . . . 538
Data items . . . . . 539
perms . . . . . 539
recipient . . . . . 539
EncryptionTypeResult . . . . . 540
Data items . . . . . 540
encryptionType . . . . . 540
FieldMDPOptionSpec . . . . . 540
Data items . . . . . 541
action . . . . . 541
fields . . . . . 541
Datatype specific settings . . . . . 541
Field Locking Action . . . . . 541
Applicable to Form Fields . . . . . 542
float . . . . . 542
Literal value . . . . . 542
Form . . . . . 542
Data items . . . . . 542
data . . . . . 543
formUrl . . . . . 543
Datatype specific settings . . . . . 543
URL . . . . . 543
Default Data . . . . . 543
Advanced settings . . . . . 543
Render Service . . . . . 543
Submit Service . . . . . 544
Schema settings . . . . . 544
FormDatatype . . . . . 545
FormModel . . . . . 545
Data items . . . . . 545
FormPreference . . . . . 545
(Deprecated) FormGuideRenderSpec . . . . . 547
Data items . . . . . 547
CB . . . . . 547
guideAccessible . . . . . 547
guideCBURL . . . . . 547
guideName . . . . . 547
guidePDF . . . . . 548
guideRSL . . . . . 548
guideStyle . . . . . 548
guideSubmitAll . . . . . 548

ifModifiedSince . . . . . 548
injectFormBridge . . . . . 548
locale . . . . . 548
Datatype specific settings . . . . . 548
If Modified Since . . . . . 548
Guide Name . . . . . 549
Guide RSL . . . . . 549
Guide PDF . . . . . 549
Guide Accessible . . . . . 549
Guide CB Url . . . . . 549
Locale . . . . . 549
Cb . . . . . 550
Guide Style . . . . . 550
Guide Submit All . . . . . 550
Inject Form Bridge . . . . . 550
FormsResult . . . . . 551
Data items . . . . . 551
action . . . . . 551
attachments . . . . . 552
charSet . . . . . 552
clickedBtn . . . . . 552
contentType . . . . . 552
formQuery . . . . . 552
locale . . . . . 553
options . . . . . 553
outputContent . . . . . 553
outputString . . . . . 553
outputType . . . . . 553
outputXML . . . . . 553
pageCount . . . . . 553
pageNumber . . . . . 553
transformationID . . . . . 553
validationErrorsList . . . . . 554
XMLData . . . . . 554
FTP FileInfo . . . . . 554
Data items . . . . . 555
directory . . . . . 555
directoryName . . . . . 555
fileName . . . . . 555
fullPath . . . . . 555
hidden . . . . . 555
readOnly . . . . . 555
size . . . . . 555
GetUsageRightsResult . . . . . 555
Data items . . . . . 555
evaluation . . . . . 556

intendedUse . . . . . 556
message . . . . . 556
notAfter . . . . . 556
notBefore . . . . . 556
profile . . . . . 556
rights . . . . . 556
useCount . . . . . 556
Group . . . . . 556
Data items . . . . . 557
canonicalName . . . . . 557
commonName . . . . . 557
domainName . . . . . 557
principalOid . . . . . 557
HashAlgorithm . . . . . 557
Datatype specific settings . . . . . 558
HTMLRenderSpec . . . . . 558
Data items . . . . . 558
cacheEnabled . . . . . 558
charset . . . . . 558
customCSSURI . . . . . 559
debugEnabled . . . . . 559
digSigCCSURI . . . . . 559
fontMapURI . . . . . 559
formModel . . . . . 559
generateTabIndex . . . . . 559
HTMLToolbar . . . . . 559
locale . . . . . 559
outputType . . . . . 560
pageNumber . . . . . 560
standAlone . . . . . 560
styleGenerationLevel . . . . . 560
toolbarURI . . . . . 560
XCIURI . . . . . 560
XMLData . . . . . 561
Datatype specific settings . . . . . 561
HTML Output Type . . . . . 561
Character Set . . . . . 561
Start Page Number . . . . . 561
Character Set . . . . . 562
Locale . . . . . 562
Cache Form On Server . . . . . 562
Populate XML Data . . . . . 562
Stand Alone Rendition . . . . . 563
Form Model . . . . . 563
XCI URI . . . . . 563
Digital Signature CSS URI . . . . . 564

Custom CSS URI . . . . . 564
HTML Toolbar . . . . . 564
HTML Toolbar URI . . . . . 564
Generate Tab Index . . . . . 564
Style Generation Level . . . . . 565
HTMLToolbar . . . . . 565
Data items . . . . . 565
IdentityDetails . . . . . 566
Data items . . . . . 566
certificate . . . . . 566
commonName . . . . . 566
country . . . . . 566
DNQualifier . . . . . 566
email . . . . . 566
generationQualifier . . . . . 566
givenName . . . . . 566
initials . . . . . 566
locality . . . . . 566
organization . . . . . 567
organizationalUnit . . . . . 567
pseudonym . . . . . 567
serialNumber . . . . . 567
state . . . . . 567
subjectName . . . . . 567
surName . . . . . 567
title . . . . . 567
IdentityInformation . . . . . 567
Data items . . . . . 567
paths . . . . . 568
policyQualifierList . . . . . 568
signerDetails . . . . . 568
status . . . . . 568
IdentityStatus . . . . . 568
IGetLinkedLCAssetsLocationResult . . . . . 568
Data items . . . . . 569
linkedLCAssetsFolderPath . . . . . 569
linkedLCAssetsName . . . . . 569
linkedLCAssetsURL . . . . . 569
ILoginSettings.LoginMode . . . . . 569
Literal value . . . . . 569
InitiatorTO . . . . . 570
int . . . . . 570
Literal value . . . . . 570
Example . . . . . 570
lc7form . . . . . 570
list . . . . . 571

Locale . . . 571
Data items . . . 571
country . . . 571
displayCountry . . . 571
display Language . . . 571
displayName . . . 571
displayVariant . . . 571
ISO3Country . . . 571
ISO3Language . . . 571
language . . . 572
variant . . . 572
LoginMode . . . 572
Datatype specific settings . . . 572
loginSettings . . . 572
Data items . . . 572
loginMode . . . 573
hostName . . . 573
userName . . . 573
password . . . 573
domainName . . . 573
long . . . 573
Literal value . . . 573
Example . . . 573
map . . . 574
MessageFormatEnum . . . 574
MDPPermissions . . . 574
Datatype specific settings . . . 574
ModeratorTO . . . 575
object . . . 575
OCSPOptionSpec . . . 575
Data items . . . 575
allowOCSPNoCheck . . . 576
doSignRequest . . . 576
goOnline . . . 576
ignoreValidityDates . . . 576
maxClockSkew . . . 576
ocspServerURL . . . 576
requestSignerCredentialAlias . . . 576
RequireOCSPCertHash . . . 576
responseFreshness . . . 577
revocationCheckStyle . . . 577
sendNonce . . . 577
URLtoConsultOption . . . 577
Datatype specific settings . . . 578
URL to Consult Option . . . 578
OCSP Server URL . . . 578

Revocation Check Style . . . . . 578
Max Clock Skew Time (minutes) . . . . . 579
Response Freshness Time (minutes) . . . . . 579
Send Nonce . . . . . 579
Sign OCSP Request . . . . . 579
Request Signer Credential Alias . . . . . 579
Go Online for OCSP . . . . . 579
Ignore Validity Dates . . . . . 579
Allow OCSP NoCheck Extension . . . . . 580
Require OCSP ISIS-MTT CertHash Extension . . . . . 580
OutputJog . . . . . 580
Data items . . . . . 580
Datatype specific settings . . . . . 580
Default Value . . . . . 580
OutputResult . . . . . 581
Data items . . . . . 581
generateDoc . . . . . 581
metaDataDoc . . . . . 581
recordLevelMetaDataList . . . . . 581
statusDoc . . . . . 581
Output Type . . . . . 581
OutputType . . . . . 582
Owner . . . . . 582
Data items . . . . . 582
businessCalendarKey . . . . . 582
disabled . . . . . 582
familyName . . . . . 582
givenName . . . . . 582
initials . . . . . 582
locale . . . . . 583
postalAddress . . . . . 583
telephoneNumber . . . . . 583
timezone . . . . . 583
userid . . . . . 583
Pagination . . . . . 583
Data items . . . . . 583
Datatype specific settings . . . . . 583
Default Value . . . . . 584
ParticipantTO . . . . . 584
Data items . . . . . 584
additionalMetadata . . . . . 584
completedBy . . . . . 584
completedFromIP . . . . . 584
disposition . . . . . 584
finalComments . . . . . 584
id . . . . . 585

reviewContext . . . . . 585
status . . . . . 585
umOid . . . . . 585
Password Encryption Option Spec . . . . . 585
Data items . . . . . 585
compatibility . . . . . 585
documentOpenPassword . . . . . 586
encryptOption . . . . . 586
permissionPassword . . . . . 586
permissionRequested . . . . . 586
Datatype specific settings . . . . . 587
Compatibility . . . . . 587
All Document Contents . . . . . 588
All Contents Except Metadata . . . . . 588
Attachments Only . . . . . 588
Document Open Password . . . . . 588
Restrict Edit Of Security Settings . . . . . 588
Permissions Password . . . . . 588
Printing Allowed . . . . . 588
Changes Allowed . . . . . 588
Password Encryption Type . . . . . 588
PathValidationOptionSpec . . . . . 589
Data items . . . . . 589
anyPolicyInhibit . . . . . 589
checkAllPaths . . . . . 589
checkCABasicConstraints . . . . . 589
explicitPolicy . . . . . 590
followURIsInAIA . . . . . 590
LDAPServer . . . . . 590
policyMappingInhibit . . . . . 590
requireValidSigForChaining . . . . . 590
Datatype specific settings . . . . . 590
Require Explicit Policy . . . . . 590
Inhibit ANY Policy . . . . . 590
Check All Paths . . . . . 591
Inhibit Policy Mapping . . . . . 591
LDAP Server . . . . . 591
Follow URIs in Certificate AIA . . . . . 591
Basic Constraints Extension required in CA Certificates . . . . . 591
Require Valid Certificate Signature During chain building . . . . . 591
PDFAConformance . . . . . 591
Data items . . . . . 592
Datatype specific settings . . . . . 592
Default Value . . . . . 592
PDFAConversionOptionSpec . . . . . 592
Data items . . . . . 593

colorSpace . . . . . 593
compliance . . . . . 593
jobLevel . . . . . 593
resultLevel . . . . . 594
signatures . . . . . 594
Datatype specific settings . . . . . 595
Compliance . . . . . 595
Result Level . . . . . 595
Signatures . . . . . 595
Color Space . . . . . 595
Verify Conversion . . . . . 595
Job Log Level . . . . . 595
PDFAConversionResult . . . . . 595
Data items . . . . . 595
conversionLog . . . . . 596
isPDFA . . . . . 596
jobLog . . . . . 596
PDFADocument . . . . . 596
PDFARevisionNumber . . . . . 596
Data items . . . . . 596
Datatype specific settings . . . . . 596
Default Value . . . . . 596
PDFAValidationOptionSpec . . . . . 597
Data items . . . . . 597
allowCertificationSignatures . . . . . 597
compliance . . . . . 597
ignoreUnusedResource . . . . . 597
logLevel . . . . . 597
resultLevel . . . . . 598
Datatype specific settings . . . . . 598
Compliance . . . . . 599
Result Level . . . . . 599
Allow Certification Signatures . . . . . 599
Ignore Unused Resources . . . . . 599
Job Log Level . . . . . 599
PDFAValidationResult . . . . . 599
Data items . . . . . 599
isPDFA . . . . . 599
jobLog . . . . . 599
validationLog . . . . . 599
PDFDocumentVerificationInfo . . . . . 600
Data items . . . . . 600
description . . . . . 600
status . . . . . 601
verificationInfos . . . . . 601
PDFFormRenderSpec . . . . . 601

Data items . . . . . 602
acrobatVersion . . . . . 602
cacheEnabled . . . . . 602
charset . . . . . 602
clientCache . . . . . 602
debugEnabled . . . . . 602
formModel . . . . . 602
generateServerAppearance . . . . . 602
linearizePDF . . . . . 603
locale . . . . . 603
PDFVersion . . . . . 603
renderAtClient . . . . . 603
seedPDF . . . . . 603
standAlone . . . . . 603
taggedPDF . . . . . 604
XCIURI . . . . . 604
XMLData . . . . . 604
Datatype specific settings . . . . . 604
Character Set . . . . . 604
Locale . . . . . 604
Cache Form On Server . . . . . 605
Acrobat Version . . . . . 605
Populate XML Data . . . . . 606
Tagged PDF . . . . . 606
Linearized PDF . . . . . 606
Seed PDF . . . . . 607
Render At Client . . . . . 607
Stand Alone Rendition . . . . . 607
Form Model . . . . . 608
XCI URI . . . . . 608
Client Cache . . . . . 608
Generate Server Appearance . . . . . 609
PDFLegalWarnings . . . . . 609
Data items . . . . . 609
alternateImagesCount . . . . . 609
annotationsCount . . . . . 609
catalogHasAACount . . . . . 609
catalogHasOpenAction . . . . . 609
devDepGS_SMaskCount . . . . . 610
devDepGSBGCount . . . . . 610
devDepGSFLCount . . . . . 610
devDepGSHTCount . . . . . 610
devDepGSOPCount . . . . . 610
devDepGSTRCount . . . . . 610
devDepGSUCRCount . . . . . 610
docHasCryptFilter . . . . . 610

docHasNonSigField . . . . . 610
docHasPresentation . . . . . 610
docHasPSXObj . . . . . 611
docHasRequirementHandler . . . . . 611
docHasXFA . . . . . 611
dynamicSigAPCount . . . . . 611
externalOPIDictsCount . . . . . 611
externalStreamsCount . . . . . 611
futurePDFVersionCount . . . . . 611
goTo3DViewActionCount . . . . . 611
goToEHasF . . . . . 611
hideActions . . . . . 612
hideAnnotationActions . . . . . 612
importDataActions . . . . . 612
invalidEOF . . . . . 612
invalidFileHeader . . . . . 612
javaScriptActions . . . . . 612
launchActions . . . . . 612
legalAttestationString . . . . . 612
malformedContentStm . . . . . 612
movieActions . . . . . 613
namedAction . . . . . 613
nonEmbeddedFontsCount . . . . . 613
optionalContentPresent . . . . . 613
pageHasAA . . . . . 613
refXobjects . . . . . 613
renditionActionCount . . . . . 613
setOCGStateActions . . . . . 613
setStateActions . . . . . 613
sigFieldHasAA . . . . . 614
sigFieldHasAction . . . . . 614
soundActions . . . . . 614
trueTypeFontsCount . . . . . 614
unknownNamedAction . . . . . 614
unknownPDFContent . . . . . 614
uriActions . . . . . 614
XObjHasInterpolate . . . . . 614
PDFOutputOptionsSpec . . . . . 614
Data items . . . . . 615
charset . . . . . 615
fileURI . . . . . 615
generateManyFiles . . . . . 615
lazyloading . . . . . 615
locale . . . . . 615
lookAhead . . . . . 615
lpdURI . . . . . 615

metaDataSpecFile . . . . . 615
printerQueueName . . . . . 616
printerURI . . . . . 616
recoredIDField . . . . . 616
recordLevel . . . . . 616
recordLevelMetaData . . . . . 616
recordName . . . . . 616
rules . . . . . 616
serverPrintSpec . . . . . 616
XCIURI . . . . . 616
Datatype specific settings . . . . . 617
General . . . . . 617
Batch . . . . . 617
Rules . . . . . 618
Destination . . . . . 619
MetaData . . . . . 619
PDFPropertiesOptionSpec . . . . . 620
Data items . . . . . 620
hasAttachments . . . . . 620
hasComments . . . . . 620
isAcroForm . . . . . 620
isFillableForm . . . . . 620
isPDFDocument . . . . . 620
isPDFPackage . . . . . 620
isXFADocument . . . . . 621
queryFormType . . . . . 621
queryPDFVersion . . . . . 621
queryRequiredAcrobatVersion . . . . . 621
queryXFAVersion . . . . . 621
PDFPropertiesResult . . . . . 621
Data items . . . . . 621
formType . . . . . 621
hasAttachments . . . . . 622
hasComments . . . . . 622
isAcroForm . . . . . 622
isFillableForm . . . . . 622
isPDFDocument . . . . . 622
isPDFPackage . . . . . 622
isXFADocument . . . . . 623
locked . . . . . 623
PDFVersion . . . . . 623
queryExceptions . . . . . 623
requiredAcrobatVersion . . . . . 623
XFAVersion . . . . . 623
PDFSeedValueOptionSpec . . . . . 623
Data items . . . . . 623

addRevInfo . . . . . 624
certificateSeedValueOptions . . . . . 624
digestMethod . . . . . 626
filter . . . . . 626
filterEX . . . . . 626
flags . . . . . 626
legalAttestations . . . . . 627
mdpValue . . . . . 627
reasons . . . . . 627
subFilter . . . . . 627
subFilterEx . . . . . 628
timeStampSeed . . . . . 628
version . . . . . 628
Datatype specific settings . . . . . 628
Signature Handler Options . . . . . 629
Signature Information . . . . . 630
Signature Type . . . . . 632
Signing Certificates . . . . . 633
Issuers and Policies . . . . . 633
PDFSignature . . . . . 634
Data items . . . . . 634
certificates . . . . . 634
contents . . . . . 634
CRLs . . . . . 634
filter . . . . . 634
OCSPResponses . . . . . 635
subFilter . . . . . 635
timestamp . . . . . 635
PDFSignatureAppearanceOptionSpec . . . . . 635
Data items . . . . . 635
appearanceType . . . . . 635
graphicPDF . . . . . 636
logoOpacity . . . . . 636
logoPDF . . . . . 636
showDate . . . . . 636
showDefaultLogo . . . . . 636
showDN . . . . . 636
showLabels . . . . . 636
showLocation . . . . . 637
showName . . . . . 637
showReason . . . . . 637
textDirection . . . . . 637
Datatype specific settings . . . . . 637
Signature Type . . . . . 637
Graphic PDF Document . . . . . 638
Use Default Adobe PDF Logo . . . . . 638

Logo PDF Document . . . . . 638
Logo Opacity . . . . . 638
Text Direction . . . . . 638
Show Name . . . . . 639
Show Reason . . . . . 639
Show Distinguished Name . . . . . 639
Show Date . . . . . 639
Show Location . . . . . 639
Show Labels . . . . . 639
PDFSignatureField . . . . . 639
Data items . . . . . 639
name . . . . . 639
properties . . . . . 640
signed . . . . . 640
type . . . . . 640
visible . . . . . 640
PDFSignatureFieldProperties . . . . . 640
Data items . . . . . 640
fieldMDP . . . . . 640
seedValue . . . . . 641
Datatype specific settings . . . . . 641
Field MDP Options Spec . . . . . 641
Seed Value Options Spec . . . . . 641
Signature Handler Options . . . . . 642
Signature Information . . . . . 644
Signature Type . . . . . 645
Signing Certificates . . . . . 646
Issuers and Policies . . . . . 647
PDFSignaturePermissions . . . . . 647
PDFSignatureType . . . . . 648
PDFSignatureVerificationInfo . . . . . 648
Data items . . . . . 648
fieldName . . . . . 648
signatureProps . . . . . 648
signatureStatus . . . . . 648
signatureType . . . . . 649
signer . . . . . 650
PDFSignatureVerificationResult . . . . . 650
Data items . . . . . 650
certPaths . . . . . 650
contactInfo . . . . . 651
dateSigned . . . . . 651
legalAttestations . . . . . 651
location . . . . . 651
numRevisions . . . . . 651
permissions . . . . . 651

policyQualifierList . . . . . 652
reason . . . . . 652
revision . . . . . 652
signatureStatus . . . . . 652
signerName . . . . . 652
signerStatus . . . . . 652
signingDateTimestamped . . . . . 653
timestamp . . . . . 653
TSAInfo . . . . . 653
TSAStatus . . . . . 653
PDFUtilitySaveMode . . . . . 653
Data items . . . . . 653
required . . . . . 654
saveStyle . . . . . 654
PolicySpec . . . . . 654
Data items . . . . . 654
accessDeniedErrorMessage . . . . . 654
alternateId . . . . . 654
autoGenerated . . . . . 654
creationTime . . . . . 655
deleted . . . . . 655
description . . . . . 655
encryptionAlgorithmAndKeySize . . . . . 655
lastUpdateTime . . . . . 655
name . . . . . 655
offlineLeasePeriod . . . . . 655
owner . . . . . 655
policyId . . . . . 655
policySetName . . . . . 655
policyType . . . . . 655
policyXml . . . . . 655
principals . . . . . 655
watermarkName . . . . . 656
PositionRectangle . . . . . 656
Data items . . . . . 656
height . . . . . 656
lowerLeftX . . . . . 656
lowerLeftY . . . . . 656
width . . . . . 656
Datatype specific settings . . . . . 656
Lower Left X . . . . . 656
Lower Left Y . . . . . 657
Height . . . . . 657
Width . . . . . 657
Principal Reference . . . . . 657
Data items . . . . . 657

canonicalname . . . . . 657
commonName . . . . . 657
domainCommonName . . . . . 657
domainName . . . . . 657
email . . . . . 657
oid . . . . . 658
org . . . . . 658
principalType . . . . . 658
status . . . . . 658
system . . . . . 658
visibility . . . . . 658
PrintedOutputOptionsSpec . . . . . 659
Data items . . . . . 659
charset . . . . . 659
copies . . . . . 659
debug . . . . . 659
fileURI . . . . . 659
generateManyFiles . . . . . 659
lazyloading . . . . . 660
locale . . . . . 660
lookAhead . . . . . 660
lpdURI . . . . . 660
metaDataSpecFile . . . . . 660
outputBin . . . . . 660
outputJog . . . . . 660
pageOffsetX . . . . . 660
pageOffsetY . . . . . 660
pagination . . . . . 661
printerQueueName . . . . . 661
printerURI . . . . . 661
recordIdField . . . . . 661
recordLevel . . . . . 661
recordLevelMetaData . . . . . 661
recordName . . . . . 661
rules . . . . . 661
serverPrintSpec . . . . . 661
staple . . . . . 662
XCIURI . . . . . 662
Datatype specific settings . . . . . 662
General . . . . . 662
Batch . . . . . 663
Rules . . . . . 663
Destination . . . . . 664
Printer . . . . . 664
MetaData . . . . . 666
PrinterProtocol . . . . . 666

Data items . . . . . 666
Datatype specific settings . . . . . 667
Default Value . . . . . 667
PrintFormat . . . . . 667
Data items . . . . . 668
Datatype specific settings . . . . . 669
Default Value . . . . . 669
PSLevel . . . . . 670
ReadPermissionsResult . . . . . 670
Data items . . . . . 671
accessPermissions . . . . . 671
inheritParentPermissions . . . . . 671
ReaderExtensionsOptionSpec . . . . . 671
Data items . . . . . 671
message . . . . . 671
modeFinal . . . . . 671
usageRights . . . . . 671
Datatype specific settings . . . . . 672
Basic Form Fill-in . . . . . 673
Import and Export Form Data . . . . . 673
Submit Outside Web Browser . . . . . 673
Database and Web Service Connectivity . . . . . 673
Add, Delete, and Change Form Fields . . . . . 673
Create Pages From Templates . . . . . 673
2D Barcode Decoding . . . . . 673
Digital Signatures . . . . . 673
Commenting . . . . . 673
Online Commenting . . . . . 673
Embedded File Attachments . . . . . 674
Draft . . . . . 674
Reader Message . . . . . 674
Reason . . . . . 674
ReceiveProtocolEnum . . . . . 674
Datatype specific settings . . . . . 674
RedactionOptionSpec . . . . . 675
Data items . . . . . 675
redactXObjectReferences . . . . . 675
redactWholeImageForUnsupportedFilter . . . . . 675
RedactionResult . . . . . 675
Data items . . . . . 675
document . . . . . 675
redactionStatus . . . . . 675
RelationInfo . . . . . 675
Data items . . . . . 676
relationDesc . . . . . 676
relationType . . . . . 676

sourceDocInfo . . . . . 676
targetDocInfo . . . . . 676
ReminderTO . . . . . 676
RenderAtClient . . . . . 676
Data items . . . . . 676
RenderOptionsSpec-FormsService . . . . . 677
Data items . . . . . 677
acrobatVersion . . . . . 677
cacheEnabled . . . . . 677
CB . . . . . 677
charset . . . . . 677
clientCache . . . . . 678
clientFrame . . . . . 678
debugEnabled . . . . . 678
digSigCSSURI . . . . . 678
exportDataFormat . . . . . 678
fontMapURI . . . . . 679
formModel . . . . . 679
guideAccessible . . . . . 679
guideCBURL . . . . . 679
guideName . . . . . 679
guidePDF . . . . . 679
guideRSL . . . . . 679
guideStyle . . . . . 679
guideSubmitAll . . . . . 680
ifModifiedSince . . . . . 680
imageURL . . . . . 680
injectFormBridge . . . . . 680
internalOptionMap . . . . . 680
linearizedPDF . . . . . 680
locale . . . . . 680
options . . . . . 680
outputType . . . . . 681
pageNumber . . . . . 681
PDF2XDP . . . . . 681
PDFVersion . . . . . 681
propertyMap . . . . . 681
re2DBarcode . . . . . 682
reCommenting . . . . . 682
reCreatePages . . . . . 682
reCredentialAlias . . . . . 682
reCredentialPassword . . . . . 682
reDigSig . . . . . 682
reEmbeddedAttachments . . . . . 683
reExpImp . . . . . 683
reFillIn . . . . . 683

reFormFieldMod . . . . . 683
renderAtClient . . . . . 683
reOnlineCommenting . . . . . 683
reOnlineForms . . . . . 684
reReaderMessage . . . . . 684
reStandaloneSubmit . . . . . 684
rootLocale . . . . . 684
seedPDF . . . . . 684
serviceId . . . . . 684
standAlone . . . . . 684
taggedPDF . . . . . 685
toolbarMenu . . . . . 685
validationBorder . . . . . 685
validationReporting . . . . . 685
validationUI . . . . . 686
XCI . . . . . 686
XCIURI . . . . . 688
XDCURI . . . . . 688
XMLData . . . . . 688
Datatype specific settings . . . . . 688
HTML Output Type . . . . . 688
Character Set . . . . . 689
Locale . . . . . 689
Display Validation Messages . . . . . 689
Validation Reporting . . . . . 690
Validation Border . . . . . 691
Populate XML Data . . . . . 691
PDF to XDP . . . . . 691
Export Data Format . . . . . 691
XCI URI . . . . . 692
RenderOptionsSpec-OutputService . . . . . 692
Data items . . . . . 692
cacheEnabled . . . . . 692
debugEnabled . . . . . 692
linearizedPDF . . . . . 693
options . . . . . 693
PDFAAmendment . . . . . 693
PDFAConformance . . . . . 693
PDFARevisionNumber . . . . . 693
pdfVersion . . . . . 693
renderAtClient . . . . . 694
retainSignatureField . . . . . 694
taggedPDF . . . . . 694
Datatype specific settings . . . . . 694
Acrobat Version . . . . . 694
Tagged PDF . . . . . 695

Linearized PDF . . . . . 695
Render At Client . . . . . 696
Retain Signature Field . . . . . 696
Debug Enabled . . . . . 696
PDF/A Revision Number . . . . . 697
PDF/A Conformance . . . . . 697
RetainSignatureField . . . . . 697
Data items . . . . . 697
Datatype specific settings . . . . . 698
Default value . . . . . 698
ReviewContextTO . . . . . 698
Data items . . . . . 698
additionalMetadata . . . . . 698
arsProcessName . . . . . 699
auditLevel . . . . . 699
changeDescription . . . . . 699
commentServerPath . . . . . 699
commentVisibility . . . . . 699
complianceCode . . . . . 699
currentRevision . . . . . 699
currentStage . . . . . 699
customAttributes . . . . . 699
id . . . . . 699
initiator . . . . . 699
invocationId . . . . . 699
purpose . . . . . 700
reviewId . . . . . 700
reviewType . . . . . 700
revision . . . . . 700
rtsProcessName . . . . . 700
stageList . . . . . 700
status . . . . . 700
stp . . . . . 700
supportingDocumentList . . . . . 700
templateAuthor . . . . . 700
templateDescription . . . . . 700
templateName . . . . . 701
title . . . . . 701
ReviewStageTO . . . . . 701
Data items . . . . . 701
additionalMetadata . . . . . 701
assignTaskToInitiatorProcess . . . . . 701
assignTaskToParticipantProcess . . . . . 701
disposition . . . . . 701
duration . . . . . 701
durationUnit . . . . . 701

endDate . . . . . 701
id . . . . . 702
name . . . . . 702
nonExpiringStage . . . . . 702
participants . . . . . 702
postProcessHookName . . . . . 702
preProcessHookName . . . . . 702
reminders . . . . . 702
reviewContext . . . . . 702
signatureType . . . . . 702
stageNo . . . . . 702
startDate . . . . . 702
status . . . . . 702
taskType . . . . . 703
type . . . . . 703
waitForExpiry . . . . . 703
ReviewTemplateTO . . . . . 703
Data items . . . . . 703
active . . . . . 703
author . . . . . 703
complianceCode . . . . . 703
customAttributes . . . . . 704
description . . . . . 704
id . . . . . 704
name . . . . . 704
RevocationCheckStyle . . . . . 704
Datatype specific settings . . . . . 704
RevocationInformation . . . . . 705
Data items . . . . . 705
data . . . . . 705
source . . . . . 705
status . . . . . 705
statusMessage . . . . . 706
type . . . . . 706
validFrom . . . . . 707
validTo . . . . . 707
rights . . . . . 707
Data items . . . . . 707
RMInspectResult . . . . . 708
Data items . . . . . 708
alternateLicenseId . . . . . 708
docName . . . . . 708
licenseId . . . . . 708
licenseIssuingAuthority . . . . . 709
policyId . . . . . 709
policyName . . . . . 709

policySetId . . . . . 709
policySetName . . . . . 709
policyType . . . . . 709
publisherId . . . . . 709
publisherName . . . . . 709
publishTime . . . . . 709
revokeURL . . . . . 709
SearchQueryResultType . . . . . 709
Data Items . . . . . 710
searchDocuments . . . . . 710
searchCount . . . . . 710
totalAvailable . . . . . 710
searchStatus . . . . . 710
Rule . . . . . 710
Data items . . . . . 710
mPattern . . . . . 710
mForm . . . . . 710
short . . . . . 710
Literal value . . . . . 710
Example . . . . . 711
SignatureProperties . . . . . 711
Data items . . . . . 711
contactInfo . . . . . 711
legalAttestations . . . . . 711
location . . . . . 711
reason . . . . . 711
revisionNumber . . . . . 711
signerName . . . . . 711
signingDate . . . . . 711
timestamp . . . . . 711
totalRevisions . . . . . 712
SignatureType . . . . . 712
Data items . . . . . 712
permissions . . . . . 712
type . . . . . 712
Staple . . . . . 712
Data items . . . . . 712
Datatype specific settings . . . . . 713
Default Value . . . . . 713
string . . . . . 713
Datatype definition . . . . . 713
Datatype specific settings . . . . . 714
StyleGenerationLevel . . . . . 714
Data items . . . . . 714
Datatype specific settings . . . . . 714
Default value . . . . . 714

TaskContext . . . . . 715
Data Items . . . . . 715
actionInstanceId . . . . . 715
assignedUser . . . . . 715
assignedUserId . . . . . 715
classOfTask . . . . . 715
createTime . . . . . 716
description . . . . . 716
formURL . . . . . 716
inputDocument . . . . . 716
isDraft . . . . . 716
isTaskActive . . . . . 716
processInstanceId . . . . . 716
processName . . . . . 717
routeList . . . . . 717
runtimeMap . . . . . 717
selectedRoute . . . . . 717
serverReplyEmail . . . . . 717
stepName . . . . . 718
taskId . . . . . 718
taskInstructions . . . . . 718
taskStatus . . . . . 718
variationInputs . . . . . 718
variationName . . . . . 719
Task Date . . . . . 719
Data items . . . . . 719
Days . . . . . 719
Hours . . . . . 720
Minutes . . . . . 720
Total Minutes . . . . . 720
UseBusinessDays . . . . . 720
Datatype specific settings . . . . . 720
Task Deadline . . . . . 720
Data items . . . . . 721
dateObj . . . . . 721
changeInstructions . . . . . 721
dateObj . . . . . 721
deadlineInstructions . . . . . 721
followRouteOnDeadline . . . . . 721
selected . . . . . 721
selectedRoute . . . . . 721
Datatype specific settings . . . . . 721
Task Delegate and Consult . . . . . 722
Data items . . . . . 722
canConsultOnlytoGroup . . . . . 722
canConsultTask . . . . . 722

canForwardOnlytoGroup . . . . . 722
canForwardTask . . . . . 722
consultNfo . . . . . 722
forwardNfo . . . . . 722
Datatype specific settings . . . . . 723
Task Priority . . . . . 723
Data items . . . . . 723
prioritySelection . . . . . 723
Datatype specific settings . . . . . 724
Task Reminder . . . . . 724
Data items . . . . . 724
changeInstructions . . . . . 724
Reminder . . . . . 724
reminderInstructions . . . . . 724
repeatSelected . . . . . 724
repeatReminder . . . . . 724
selected . . . . . 725
Datatype specific settings . . . . . 725
Task Result . . . . . 725
Data items . . . . . 725
actionInstanceId . . . . . 725
assignedDate . . . . . 725
attachmentList . . . . . 725
completedBy . . . . . 725
completedById . . . . . 725
completedDate . . . . . 725
completionNotes . . . . . 726
formDataDocument . . . . . 726
formDataXML . . . . . 726
initialPrincipal . . . . . 726
initialPrincipalId . . . . . 726
ipAddress . . . . . 726
isActive . . . . . 726
isCompleted . . . . . 726
isDeadlined . . . . . 726
isTerminated . . . . . 726
possibleUserActions . . . . . 726
selectedUserAction . . . . . 727
stepName . . . . . 727
taskId . . . . . 727
taskStatus . . . . . 727
Task Result Collection . . . . . 727
Data items . . . . . 728
size . . . . . 728
taskResults . . . . . 728
userActions . . . . . 728

Task Routes and Priority . . . . . 728
Data items . . . . . 728
initTaskWithRoute . . . . . 728
mustSelectRouteName . . . . . 728
prioritySelection . . . . . 728
Datatype specific settings . . . . . 729
Task Runtime UI . . . . . 729
Data items . . . . . 729
approvalContainerUI . . . . . 729
customUI . . . . . 729
mustOpenFormToComplete . . . . . 730
openFormFullScreen . . . . . 730
tloPath . . . . . 730
type . . . . . 730
Datatype specific settings . . . . . 730
Task User Selection . . . . . 730
Data items . . . . . 731
domainId . . . . . 731
canonicalName . . . . . 731
All other data items . . . . . 731
Datatype specific settings . . . . . 731
TemplateSearchFilter . . . . . 731
Data items . . . . . 732
author . . . . . 732
complianceCode . . . . . 732
customAttributes . . . . . 732
keywords . . . . . 732
pageNumber . . . . . 732
pageSize . . . . . 732
retrieveActiveOnly . . . . . 732
templateName . . . . . 732
Timestamp . . . . . 733
Data items . . . . . 733
timestamp . . . . . 733
timestamped . . . . . 733
timestampInformation . . . . . 733
Timezone . . . . . 733
Data items . . . . . 733
displayName . . . . . 733
DSTSavings . . . . . 733
ID . . . . . 733
rawOffset . . . . . 734
ToImageOptionsSpec . . . . . 734
Data items . . . . . 734
cmykPolicy . . . . . 734
colorCompression . . . . . 734

colorSpace . . . . . 734
filter . . . . . 735
format . . . . . 735
grayScaleCompression . . . . . 735
grayScalePolicy . . . . . 735
imageConvertFormat . . . . . 736
interlace . . . . . 736
monochrome . . . . . 736
options . . . . . 736
resolution . . . . . 736
rgbPolicy . . . . . 736
rowsPerStrip . . . . . 737
tileSize . . . . . 737
ToPSOptionsSpec . . . . . 737
Data items . . . . . 737
allowBinaryContent . . . . . 737
bleedMarks . . . . . 737
color . . . . . 737
colorBars . . . . . 738
convertTrueTypeToType1 . . . . . 738
emitCIDFontType2 . . . . . 738
emitPSFormObjects . . . . . 738
expandToFit . . . . . 738
fontInclusion . . . . . 738
includeComments . . . . . 739
lineWeight . . . . . 739
pageInformation . . . . . 739
pageRange . . . . . 739
pageSize . . . . . 739
pageSizeHeight . . . . . 739
pageSizeWidth . . . . . 740
psLevel . . . . . 740
registrationMarks . . . . . 740
resolution . . . . . 740
reverse . . . . . 740
rotateAndCenter . . . . . 740
shrinkToFit . . . . . 740
style . . . . . 740
trimMarks . . . . . 741
useMaxJPEGImageResolution . . . . . 741
ToSWFOptionsSpec . . . . . 741
Data items . . . . . 741
setHeight . . . . . 741
setWidth . . . . . 741
setVersion . . . . . 741
setSampleX . . . . . 742

setSampleY . . . . . 742
setProcessSignatures . . . . . 742
setPageRange . . . . . 742
TransferModeEnum . . . . . 742
TransformationFormat . . . . . 742
Data items . . . . . 742
Datatype specific settings . . . . . 743
Default value . . . . . 743
TransformTo . . . . . 743
Data items . . . . . 743
Datatype specific settings . . . . . 744
Default value . . . . . 744
TransportSecurityEnum . . . . . 745
Datatype specific settings . . . . . 746
TSPOptionSpec . . . . . 746
Data items . . . . . 746
tspHashAlgorithm . . . . . 746
tspRevocationCheckStyle . . . . . 747
SendNonce . . . . . 747
tspServerPassword . . . . . 747
tspServerURL . . . . . 747
tspServerUserName . . . . . 747
tspSize . . . . . 747
useExpiredTimestamps . . . . . 748
Datatype specific settings . . . . . 748
Time Stamp Server URL . . . . . 748
Time Stamp Server Username . . . . . 748
Time Stamp Server Password . . . . . 748
Time Stamp Server Hash Algorithm . . . . . 748
Revocation Check Style . . . . . 748
Use Expired Timestamps . . . . . 749
Predicted Time Stamp Token Size (In Bytes) . . . . . 749
Send Nonce . . . . . 749
UpdateVersionType . . . . . 749
Literal value . . . . . 749
URLSpec . . . . . 750
Data items . . . . . 750
applicationWebRoot . . . . . 750
baseURL . . . . . 750
contentRootURI . . . . . 750
options . . . . . 751
targetURL . . . . . 751
Datatype specific settings . . . . . 751
Application Web Root . . . . . 751
Target URL . . . . . 751
Content Root URI . . . . . 752

Base URL . . . 752
User . . . 752
Data items . . . 752
canonicalName . . . 753
commonName . . . 753
domainName . . . 753
email . . . 753
familyName . . . 753
givenName . . . 753
initials . . . 753
org . . . 753
postalAddress . . . 753
principalOid . . . 753
telephoneNumber . . . 753
userId . . . 753
User Action Name . . . 754
Data items . . . 754
confirmation . . . 754
confirmationMessage . . . 754
destination . . . 754
destinationId . . . 754
name . . . 754
routeId . . . 754
routeName . . . 755
toolTip . . . 755
userActionName . . . 755
Userlist . . . 755
UserPreferenceTO . . . 755
Data items . . . 755
notificationEnabled . . . 755
savedSearches . . . 755
umOid . . . 755
VerificationTime . . . 756
Datatype specific settings . . . 756
xfaForm . . . 757
Data items . . . 757
data . . . 757
templateUrl . . . 757
Datatype specific settings . . . 757
Template Url . . . 758
Default Data . . . 758
Advanced settings . . . 758
Render Service . . . 758
Submit Service . . . 759
Schema Settings . . . 760
xml . . . 760

Literal value . . . . . 761
Example . . . . . 761
Datatype specific settings . . . . . 761
XMLFormat . . . . . 761
XMLSignatureVerificationResult . . . . . 762
Data items . . . . . 762
certificateList . . . . . 762
dateSigned . . . . . 762
signatureStatus . . . . . 762
signerCert . . . . . 762
signerName . . . . . 762
signerStatus . . . . . 762
XMPUtilityMetadata . . . . . 762
Data items . . . . . 763
author . . . . . 763
creator . . . . . 763
keywords . . . . . 763
producer . . . . . 763
subject . . . . . 763
title . . . . . 763
Find Type . . . . . 763

**Chapter: 22** **Service reference . . . . . .764**

Configuring for proxy servers . . . . . 764
Configuring Workbench . . . . . 764
Modify the workbench.ini file: . . . . . 765
Configuring the AEM Forms Server . . . . . 765
Digital signatures and proxy servers . . . . . 766
About deprecated operations . . . . . 766
Common operation properties . . . . . 766
General . . . . . 766
Name . . . . . 766
Description . . . . . 767
Category . . . . . 767
Service Name . . . . . 767
Service Operation . . . . . 767
Route Evaluation . . . . . 767
Process services . . . . . 767
invoke . . . . . 767
Input . . . . . 767
Output . . . . . 768
Invocation Policy . . . . . 768
Services not for use in processes . . . . . 768
About Select Asset . . . . . 769
Select an asset: . . . . . 769
Filtering operation properties . . . . . 769

Assembler . . . . . 769
Assembler DDX Editor . . . . . 770
Rights Management and Assembler . . . . . 770
invokeDDX operation . . . . . 770
Input properties . . . . . 771
Output properties . . . . . 773
Exceptions . . . . . 774
invokeDDXOneDocument operation . . . . . 774
Input properties . . . . . 774
Output properties . . . . . 775
Exceptions . . . . . 776
Assembler exceptions . . . . . 776
OperationException . . . . . 776
Asset Placement . . . . . 776
Generate Interactive Document operation . . . . . 776
Input properties . . . . . 776
Caching properties . . . . . 777
Output . . . . . 778
Exceptions . . . . . 778
Barcoded forms . . . . . 778
Decode operation . . . . . 778
Input properties . . . . . 778
Output properties . . . . . 780
Exceptions . . . . . 780
Extract to XML operation . . . . . 780
Input properties . . . . . 780
Output properties . . . . . 781
Exceptions . . . . . 781
barcoded forms exceptions . . . . . 781
DecodingException . . . . . 782
Batch Processor . . . . . 782
Get Batch Job operation . . . . . 782
Input properties . . . . . 782
Output property . . . . . 782
Exceptions . . . . . 783
List Batch Jobs operation . . . . . 783
Input properties . . . . . 783
Output property . . . . . 783
Exceptions . . . . . 783
List Batch Job Steps operation . . . . . 783
Input properties . . . . . 783
Output property . . . . . 784
Exceptions . . . . . 784
List Running Batch Jobs operation . . . . . 784
Input properties . . . . . 784
Output property . . . . . 784

Exceptions . . . . . 784
Purge Batch Repository operation . . . . . 784
Input properties . . . . . 785
Exceptions . . . . . 785
Restart Batch Job operation . . . . . 785
Input properties . . . . . 785
Output property . . . . . 785
Exceptions . . . . . 785
Run Flat File Job operation . . . . . 786
Job Identification properties . . . . . 786
Service Operation properties . . . . . 786
Static Service Parameters properties . . . . . 786
Batch Input . . . . . 786
Batch Output . . . . . 787
Optimization properties . . . . . 787
Result properties . . . . . 788
Exceptions . . . . . 788
Run JDBC Cursor Job operation . . . . . 788
Job Identification properties . . . . . 788
Service Operation properties . . . . . 788
Static Service Parameters properties . . . . . 789
Batch Input properties . . . . . 789
Batch Output properties . . . . . 789
Optimization properties . . . . . 789
Result properties . . . . . 790
Exceptions . . . . . 790
Run JDBC Paging Job operation . . . . . 790
Job Identification properties . . . . . 790
Service Operation properties . . . . . 791
Static Service Parameters properties . . . . . 791
Batch Input properties . . . . . 791
Batch Output properties . . . . . 792
Optimization properties . . . . . 792
Result properties . . . . . 792
Exceptions . . . . . 793
Run XML File Job operation . . . . . 793
Job Identification properties . . . . . 793
Service Operation properties . . . . . 793
Static Service Parameters properties . . . . . 793
Batch Input properties . . . . . 793
Batch Output properties . . . . . 794
Optimization properties . . . . . 794
Result properties . . . . . 795
Exceptions . . . . . 795
Stop Batch Job operation . . . . . 795
Input properties . . . . . 795

Output property . . . . . 795
Exceptions . . . . . 795
Central Migration Bridge . . . . . 795
Central Migration Bridge service configuration . . . . . 796
centralDataAccess operation . . . . . 796
Input properties . . . . . 796
Output properties . . . . . 797
Exceptions . . . . . 797
centralMerge operation . . . . . 797
Input properties . . . . . 797
Output properties . . . . . 798
centralTransformation operation . . . . . 799
Input properties . . . . . 799
Output properties . . . . . 800
Exceptions . . . . . 801
centralXMLImport operation . . . . . 801
Input properties . . . . . 801
Output properties . . . . . 802
Exceptions . . . . . 802
Central Migration Bridge exception . . . . . 802
CentralMigrationBridgeException . . . . . 802
Content Repository Connector for EMC Documentum . . . . . 803
Content Repository Connector for EMC Documentum service configuration
803
Create folders operation . . . . . 803
Login Settings properties . . . . . 803
Folder Creation Settings properties . . . . . 804
Result properties . . . . . 805
Exceptions . . . . . 805
Create folders with type and attributes operation . . . . . 805
Login Settings properties . . . . . 805
Folder Creation Settings properties . . . . . 806
Folder Type and Meta-Data properties . . . . . 807
Result properties . . . . . 807
Exceptions . . . . . 808
Create Relationship operation . . . . . 808
Login Settings properties . . . . . 808
Relationship Creation Settings properties . . . . . 809
Exceptions . . . . . 810
Delete Content operation . . . . . 810
Login Settings properties . . . . . 810
Document/Folder Settings properties . . . . . 811
Exceptions . . . . . 812
Execute DQL Query operation . . . . . 812
Login Settings properties . . . . . 812
Repository Settings properties . . . . . 813

Query Settings properties . . . . . 813
Result properties . . . . . 813
Exceptions . . . . . 814
Get Linked LC Assets location operation . . . . . 814
Login Settings properties . . . . . 814
ECM Object Settings properties . . . . . 815
Relationship Type Settings properties . . . . . 815
Results properties . . . . . 815
Exceptions . . . . . 816
Get Related operation . . . . . 816
Login Settings properties . . . . . 816
Relationship Creation Settings properties . . . . . 817
Result properties . . . . . 818
Exceptions . . . . . 818
Retrieve Content operation . . . . . 818
Login Settings properties . . . . . 818
Document Settings properties . . . . . 819
Meta-Data properties . . . . . 820
Results properties . . . . . 820
Exceptions . . . . . 822
Set Link To LC Assets operation . . . . . 822
Login Settings properties . . . . . 822
ECM Object Settings properties . . . . . 823
Form Template Settings properties . . . . . 823
Relationship Settings properties . . . . . 824
Exceptions . . . . . 824
Store Content operation . . . . . 824
Login Settings properties . . . . . 824
Document Creation Settings properties . . . . . 825
Document Content Settings properties . . . . . 826
Meta-Data properties . . . . . 827
Results properties . . . . . 827
Exceptions . . . . . 828
Content Repository Connector for EMC Documentum exceptions 828
RepositoryException . . . . . 828
Content Repository Connector for IBM Content Manager . . . . . 828
Create item operation . . . . . 828
Login Settings properties . . . . . 829
Item Creation Settings properties . . . . . 830
Meta-Data properties . . . . . 830
Result properties . . . . . 831
Exceptions . . . . . 831
Create Folder operation . . . . . 832
Login Settings properties . . . . . 832
Folder Creation Settings properties . . . . . 833
Meta-Data properties . . . . . 833

Results properties . . . . . 834
Exceptions . . . . . 834
Create Relationship operation . . . . . 834
Login Settings properties . . . . . 835
Create Relationship Settings properties . . . . . 836
Result properties . . . . . 837
Exceptions . . . . . 837
Delete Item operation . . . . . 837
Login Settings properties . . . . . 837
Delete Item Details properties . . . . . 839
Exceptions . . . . . 839
Get Linked LC Assets Location operation . . . . . 839
Login Settings properties . . . . . 839
Form Data Settings properties . . . . . 841
Relationship Data Settings properties . . . . . 841
Output properties . . . . . 841
Exceptions . . . . . 842
Get Related Items operation . . . . . 842
Login Settings properties . . . . . 842
Get Related Items Settings properties . . . . . 843
Results properties . . . . . 844
Exceptions . . . . . 844
Retrieve Item operation . . . . . 844
Login Settings properties . . . . . 844
Retrieve Item Settings properties . . . . . 846
Meta-Data properties . . . . . 846
Results properties . . . . . 846
Exceptions . . . . . 847
Search Items operation . . . . . 847
Login Settings properties . . . . . 847
Search Item Settings properties . . . . . 849
Result properties . . . . . 849
Exceptions . . . . . 849
Set Link To LC Assets operation . . . . . 849
Login Settings . . . . . 850
Form Data and Relationship Settings properties . . . . . 851
Form Template and ALO Settings properties . . . . . 851
Meta-Data properties . . . . . 852
Exceptions . . . . . 852
Update Item operation . . . . . 852
Login Settings properties . . . . . 852
Item Update Settings properties . . . . . 854
Meta-Data properties . . . . . 854
Results properties . . . . . 855
Exceptions . . . . . 856
Content Repository Connector for IBM Content Manager exceptions 856

RepositoryException . . . . . 856
Content Repository Connector for IBM FileNet . . . . . 856
Content Repository Connector for IBM FileNet service configuration 856
Create folders operation . . . . . 857
Login Settings properties . . . . . 857
Object Store and Folder Creation Settings properties . . . . . 858
Result properties . . . . . 858
Exceptions . . . . . 858
Create Relationship operation . . . . . 858
Login Settings properties . . . . . 858
Relationship Creation Settings properties . . . . . 859
Meta-Data properties . . . . . 860
Exceptions properties . . . . . 860
Delete Content opreation . . . . . 861
Login Settings properties . . . . . 861
Document/Folder Settings properties . . . . . 862
Exceptions . . . . . 862
Get Linked LC Assets Location operation . . . . . 862
Login Settings properties . . . . . 862
Object Store and Form Data Settings properties . . . . . 863
Relationship Type Settings properties . . . . . 864
Output properties . . . . . 864
Exceptions . . . . . 864
Get Related operation . . . . . 864
Login Settings properties . . . . . 864
Relationship Information properties . . . . . 865
Result properties . . . . . 866
Exceptions . . . . . 866
Retrieve Content operation . . . . . 866
Login Settings properties . . . . . 866
Document Settings properties . . . . . 867
Meta-Data properties . . . . . 868
Results properties . . . . . 868
Exceptions . . . . . 870
Set Link to LC Assets operation . . . . . 870
Login Settings properties . . . . . 870
Object Store and ECM Object Settings properties . . . . . 871
Form Template Settings properties . . . . . 871
Relationship Settings properties . . . . . 871
Exceptions . . . . . 872
Store Content operation . . . . . 872
Login Settings properties . . . . . 872
Object Store and Folder Settings properties . . . . . 873
Document Creation Settings properties . . . . . 874
Document Content Settings properties . . . . . 874
Meta-Data properties . . . . . 874

Results properties . . . . . 875
Exceptions . . . . . 876
Content Repository Connector for IBM FileNet exceptions . . . 876
RepositoryException . . . . . 876
Connector for Microsoft SharePoint . . . . . 876
Connector for Microsoft SharePoint service configuration . . . 876
Create Folder operation . . . . . 876
Login Settings properties . . . . . 876
Site and Document Library Information properties . . . . . 877
Folder Creation Settings properties . . . . . 878
Result properties . . . . . 878
Exceptions . . . . . 878
Create Document operation . . . . . 878
Login Settings properties . . . . . 878
Site and Document Library Information properties . . . . . 879
Document Content Information properties . . . . . 880
Document Creation Settings properties . . . . . 880
Properties . . . . . 880
Result properties . . . . . 881
Exceptions . . . . . 881
Check In File operation . . . . . 881
Login Settings properties . . . . . 881
Site and Document Library Information properties . . . . . 882
File Information properties . . . . . 882
Version Update Settings properties . . . . . 882
Result properties . . . . . 883
Exceptions . . . . . 883
Check Out File operation . . . . . 883
Login Settings properties . . . . . 883
Site and Document Library Information properties . . . . . 884
File Information properties . . . . . 884
Result properties . . . . . 885
Exceptions . . . . . 885
Cancel File Check Out operation . . . . . 885
Login Settings properties . . . . . 885
Site and Document Library Information properties properties . 886
File Location Information properties . . . . . 886
Result properties . . . . . 887
Exceptions . . . . . 887
Search operation . . . . . 887
Login Settings properties . . . . . 887
Context Scope Information properties . . . . . 888
Custom Scope Information properties properties . . . . . 888
Query Creation Settings properties . . . . . 888
Result properties . . . . . 889
Exceptions . . . . . 889

Delete operation . . . . . 889
Login Settings properties . . . . . 889
Site and Document Library Information properties . . . . . 890
File/Folder Location Settings properties . . . . . 890
Exceptions . . . . . 891
Get Properties operation . . . . . 891
Login Settings properties . . . . . 891
Site and Document Library Information properties . . . . . 892
File Location Details properties . . . . . 892
Result properties . . . . . 892
Exceptions . . . . . 893
Retrieve Document Content operation . . . . . 893
Login Settings properties . . . . . 893
Site and Document Library Information properties properties . 894
Document Identification properties . . . . . 894
Result properties . . . . . 894
Exceptions . . . . . 894
Set Document Content operation . . . . . 895
Login Settings properties . . . . . 895
Site and Document Library Information properties . . . . . 896
Document Identification properties . . . . . 896
Document Content Information properties . . . . . 896
Result . . . . . 896
Exceptions . . . . . 896
Update Properties operation . . . . . 897
Login Settings properties . . . . . 897
Site and Document Library Information . . . . . 898
Document Identification properties . . . . . 898
Document Properties properties . . . . . 898
Exceptions . . . . . 899
Connector for Microsoft SharePoint exception . . . . . 899
RepositoryException . . . . . 899
Convert PDF . . . . . 899
toImage operation . . . . . 899
Input properties . . . . . 899
Output properties . . . . . 903
Exceptions . . . . . 903
toPS2 operation . . . . . 903
Input properties . . . . . 904
Output properties . . . . . 906
Exceptions . . . . . 906
toSWF operation . . . . . 906
Input properties . . . . . 907
Output properties . . . . . 907
Exceptions . . . . . 907
Convert PDF exceptions . . . . . 907

ConvertPDFException . . . . . 907
DSCException . . . . . 908
Decision Point . . . . . 908
execute . . . . . 908
Default Render . . . . . 908
invoke operation . . . . . 909
Input properties . . . . . 909
Output properties . . . . . 910
Invocation Policy properties . . . . . 910
Distiller . . . . . 910
CreatePDF operation . . . . . 911
Input properties . . . . . 911
Conversion options properties . . . . . 911
Advanced options properties . . . . . 911
Output properties . . . . . 912
Optional Output properties . . . . . 912
Exceptions . . . . . 912
Distiller exceptions . . . . . 912
ConversionException . . . . . 913
FileFormatNotSupportedException . . . . . 913
InvalidParameterException . . . . . 913
DocConverter . . . . . 913
Validate PDF/A operation . . . . . 913
Input properties . . . . . 913
Output properties . . . . . 915
Exceptions . . . . . 915
Convert to PDF/A operation . . . . . 915
Input properties . . . . . 915
Output properties . . . . . 918
Exceptions . . . . . 918
DocConverter exceptions . . . . . 918
ConversionException . . . . . 918
ValidationException . . . . . 918
Email . . . . . 918
Email service configuration . . . . . 919
Receive operation . . . . . 920
Filter properties . . . . . 920
Folder Options . . . . . 921
Retrieved Email Contents properties . . . . . 921
From properties . . . . . 921
Retrieved Email Attributes properties . . . . . 921
Attachments properties . . . . . 922
Connection Settings properties . . . . . 923
Test properties . . . . . 924
Exceptions . . . . . 924
Send With Document operation . . . . . 924

To Addresses properties . . . . . 925
From Addresses properties . . . . . 925
Contents properties . . . . . 925
Attachments properties . . . . . 927
Connection Settings properties . . . . . 927
Exceptions . . . . . 928
Send With Map of Attachments operation . . . . . 928
To Addresses properties . . . . . 929
From Addresses properties . . . . . 929
Contents properties . . . . . 929
Attachments properties . . . . . 931
Connection Settings properties . . . . . 932
Exceptions . . . . . 933
Email exceptions . . . . . 933
ConnectionFailedException . . . . . 933
MessageNotFoundException . . . . . 933
ReceiveMailFailedException . . . . . 933
SendMailFailedException . . . . . 933
About Message Body Editor . . . . . 933
Design tab . . . . . 933
HTML Preview tab . . . . . 934
About Attachments . . . . . 934
Encryption . . . . . 935
Certificate Encrypt PDF operation . . . . . 936
Input properties . . . . . 936
Output properties . . . . . 938
Exceptions . . . . . 938
Get PDF Encryption Type operation . . . . . 938
Input properties . . . . . 938
Output properties . . . . . 939
Exceptions . . . . . 939
Password Encrypt PDF operation . . . . . 939
Input properties . . . . . 939
Output properties . . . . . 941
Exceptions . . . . . 941
Remove PDF Certificate Encryption operation . . . . . 941
Input properties . . . . . 941
Output properties . . . . . 942
Exceptions . . . . . 942
Remove PDF Password Encryption operation . . . . . 942
Input properties . . . . . 942
Output properties . . . . . 942
Exceptions . . . . . 943
Unlock Certificate Encrypted PDF operation . . . . . 943
Input properties . . . . . 943
Output properties . . . . . 943

Exceptions . . . . . 943
Unlock Password Encrypted PDF operation . . . . . 944
Input properties . . . . . 944
Output properties . . . . . 944
Exceptions . . . . . 944
Encryption exceptions . . . . . 944
EncryptionServiceException . . . . . 944
Execute Script . . . . . 945
executeScript . . . . . 945
Input operation . . . . . 946
patExecContext reference . . . . . 946
getProcessId . . . . . 946
getBranchId . . . . . 946
getActionId . . . . . 946
getExecuteTemplate . . . . . 947
getActionTemplate . . . . . 947
getProcessDataBooleanValue . . . . . 947
getProcessDataShortValue . . . . . 947
getProcessDataIntValue . . . . . 948
getProcessDataLongValue . . . . . 948
getProcessDataDoubleValue . . . . . 948
getProcessDataFloatValue . . . . . 949
getProcessDataStringValue . . . . . 949
getProcessDataDocumentValue . . . . . 949
getProcessDataListValue . . . . . 950
getProcessDataMapValue . . . . . 950
getProcessDataValue . . . . . 950
setProcessDataBooleanValue . . . . . 951
setProcessDataShortValue . . . . . 951
setProcessDataIntValue . . . . . 951
setProcessDataLongValue . . . . . 952
setProcessDataDoubleValue . . . . . 952
setProcessDataListValue . . . . . 952
appendToProcessDataListValue . . . . . 953
setProcessDataListValue . . . . . 953
removeProcessDataListValue . . . . . 953
setProcessDataMapValue . . . . . 953
appendProcessDataMapValue . . . . . 954
setProcessDataMapValue . . . . . 954
removeProcessDataMapValue . . . . . 954
setProcessDataFloatValue . . . . . 955
setProcessDataStringValue . . . . . 955
setProcessDataDocumentValue . . . . . 955
setProcessDataValue . . . . . 955
setProcessDataWithExpression . . . . . 956
getContainedDataTypeForCollection . . . . . 956

getProcessDataValueByVariableName . . . . . 957
File Utilities . . . . . 958
Delete operation . . . . . 958
Input properties . . . . . 958
Output properties . . . . . 959
Exceptions . . . . . 959
Exists operation . . . . . 959
Input properties . . . . . 959
Output properties . . . . . 959
Exceptions . . . . . 959
Find operation . . . . . 959
Input properties . . . . . 960
Output properties . . . . . 960
Exceptions . . . . . 960
Is Absolute operation . . . . . 961
Input properties . . . . . 961
Output properties . . . . . 961
Exceptions . . . . . 961
Is Directory operation . . . . . 961
Input properties . . . . . 961
Output properties . . . . . 961
Exceptions . . . . . 962
Is File operation . . . . . 962
Input properties . . . . . 962
Output properties . . . . . 962
Exceptions . . . . . 962
Is Hidden operation . . . . . 962
Input properties . . . . . 962
Output properties . . . . . 963
Exceptions . . . . . 963
Make Directory operation . . . . . 963
Input properties . . . . . 963
Output properties . . . . . 963
Exceptions . . . . . 963
Make Directory Tree operation . . . . . 964
Input properties . . . . . 964
Output properties . . . . . 964
Exceptions . . . . . 964
Read Document operation . . . . . 964
Input properties . . . . . 964
Output properties . . . . . 964
Exceptions . . . . . 965
Read String operation . . . . . 965
Input properties . . . . . 965
Output properties . . . . . 965
Exceptions . . . . . 966

Read XML operation . . . . . 966
Input properties . . . . . 966
Output properties . . . . . 967
Exceptions . . . . . 967
Rename To operation . . . . . 967
Input properties . . . . . 967
Output properties . . . . . 968
Exceptions . . . . . 968
Set Read Only operation . . . . . 968
Input properties . . . . . 968
Output properties . . . . . 969
Exceptions . . . . . 969
Write Document operation . . . . . 969
Input properties . . . . . 969
Output properties . . . . . 970
Exceptions . . . . . 971
Write String operation . . . . . 971
Input operation . . . . . 971
Output properties . . . . . 972
Exceptions . . . . . 973
Write XML operation . . . . . 973
Input properties . . . . . 973
Output properties . . . . . 975
Exceptions . . . . . 975
Regular expressions syntax . . . . . 975
Literal characters . . . . . 975
Quantifiers . . . . . 976
Character classes . . . . . 976
File Utilities exceptions . . . . . 977
FileUtilsException . . . . . 977
Form Augmenter . . . . . 977
Get value of a form field operation . . . . . 978
Input properties . . . . . 978
Output properties . . . . . 978
Exceptions . . . . . 978
Inject Form Bridge operation . . . . . 979
Input properties . . . . . 980
Output properties . . . . . 980
Exceptions . . . . . 980
Insert Workflow XFO Data operation . . . . . 980
Input properties . . . . . 981
Output properties . . . . . 982
Exceptions . . . . . 982
Lookup and Insert Workflow XFO Data operation . . . . . 982
Input properties . . . . . 983
Output properties . . . . . 983

Exceptions . . . . . 984
Remove Data Fields operation . . . . . 984
Input properties . . . . . 984
Output properties . . . . . 984
Exceptions . . . . . 984
Form Augmenter exceptions . . . . . 984
Java.lang.Exception . . . . . 984
Form Data Integration . . . . . 985
Retaining legacy appearance of PDF Forms in AEM Forms . . . 985
exportData operation . . . . . 985
Input properties . . . . . 986
Output properties . . . . . 986
Exceptions . . . . . 986
importData operation . . . . . 987
Input properties . . . . . 987
Output properties . . . . . 987
Exceptions . . . . . 987
Form Data Integration exceptions . . . . . 987
ImportFormDataException . . . . . 987
ExportFormDataException . . . . . 988
Forms . . . . . 988
Retaining legacy appearance of PDF Forms in AEM Forms . . . 988
processFormSubmission operation . . . . . 989
Input properties . . . . . 989
Form Submission Options properties . . . . . 990
Output properties . . . . . 990
Additional Output properties . . . . . 990
Exceptions properties . . . . . 991
(Deprecated) renderHTMLForm operation . . . . . 991
Input properties . . . . . 992
Template Options properties . . . . . 992
Advanced HTML Options properties . . . . . 995
Render Options properties . . . . . 996
Additional Options properties . . . . . 998
Output properties . . . . . 998
Additional Output properties . . . . . 999
Exceptions . . . . . 999
renderHTMLForm operation (deprecated) . . . . . 999
Input properties . . . . . 1000
Output properties . . . . . 1007
Exceptions . . . . . 1008
renderPDFForm operation . . . . . 1008
Input properties . . . . . 1008
Template options properties . . . . . 1009
PDF Options properties . . . . . 1010
Render options properties . . . . . 1011

Additional Options properties . . . . . 1013
Output properties . . . . . 1014
Additional Output properties . . . . . 1014
Exceptions . . . . . 1015
renderPDFForm operation(deprecated) . . . . . 1015
Input properties . . . . . 1015
Output properties . . . . . 1021
Exceptions . . . . . 1022
Forms exceptions . . . . . 1022
ProcessFormSubmissionException . . . . . 1022
RenderFormException . . . . . 1022
FTP . . . . . 1022
FTP service configuration . . . . . 1023
Delete file operation . . . . . 1023
Remote File Details properties . . . . . 1023
Connection Settings properties . . . . . 1024
Delete Operation Result properties . . . . . 1024
Exceptions . . . . . 1024
Get operation . . . . . 1024
Remote File Details properties . . . . . 1025
File Transfer Mode properties . . . . . 1025
Connection Settings properties . . . . . 1025
File Content properties . . . . . 1026
Exceptions . . . . . 1026
Get list of files operation . . . . . 1026
Remote Directory Details properties . . . . . 1026
Connection Settings properties . . . . . 1026
Retrieved Files Detail properties . . . . . 1027
Exceptions . . . . . 1027
Get multiple documents operations . . . . . 1027
Remote Files Options properties . . . . . 1027
Connection Settings properties . . . . . 1028
Retrieved Documents properties . . . . . 1029
Exceptions . . . . . 1029
Get to file system operation . . . . . 1029
Local File Details properties . . . . . 1029
Remote File Details properties . . . . . 1029
File Transfer Mode properties . . . . . 1030
Connection Settings properties . . . . . 1030
File Transfer Result properties . . . . . 1031
Exceptions . . . . . 1031
Put operation . . . . . 1031
File Details properties . . . . . 1031
Remote File Details properties . . . . . 1031
File Transfer Mode properties . . . . . 1032
Connection Settings properties . . . . . 1032

File Transfer Result properties . . . . . 1032
Exceptions . . . . . 1033
Put from file system operation . . . . . 1033
File Details properties . . . . . 1033
Remote Path properties . . . . . 1033
File Transfer Mode properties . . . . . 1033
Connection Settings properties . . . . . 1034
File Transfer Result properties . . . . . 1034
Exceptions . . . . . 1034
Put multiple documents operation . . . . . 1035
File Options properties . . . . . 1035
Connection Settings properties . . . . . 1035
Result properties . . . . . 1036
Exceptions . . . . . 1036
-FTP Exceptions . . . . . 1036
FTPConnectionException . . . . . 1036
FTPFileNotFoundException . . . . . 1036
FTPFileUnavailableException . . . . . 1037
IllegalArgumentException . . . . . 1037
IOException . . . . . 1037
Generate PDF . . . . . 1037
Generate PDF service configuration . . . . . 1038
CreatePDF operation . . . . . 1040
Input properties . . . . . 1040
Conversion Options properties . . . . . 1040
Advanced Options properties . . . . . 1041
Output properties . . . . . 1042
Optional Output properties . . . . . 1042
Exceptions . . . . . 1042
ExportPDF operation . . . . . 1042
Input properties . . . . . 1043
Output properties . . . . . 1044
Exceptions . . . . . 1044
HtmlFileToPDF operation . . . . . 1044
Input properties . . . . . 1044
Output properties . . . . . 1045
Exceptions . . . . . 1045
Html URL To PDF operation . . . . . 1045
Input properties . . . . . 1045
Output properties . . . . . 1046
Exceptions . . . . . 1047
Optimize PDF operation . . . . . 1047
Input properties . . . . . 1047
Output properties . . . . . 1048
Exceptions . . . . . 1048
Generate PDF exceptions . . . . . 1048

ConversionException . . . . . 1048
FileFormatNotSupportedException . . . . . 1048
InvalidParameterException . . . . . 1048
JDBC . . . . . 1048
JDBC service configuration . . . . . 1049
DatasourceName . . . . . 1049
Call Stored Procedure operation . . . . . 1049
Input properties . . . . . 1049
Output properties . . . . . 1049
Exceptions . . . . . 1050
Execute SQL Statement operation . . . . . 1050
Input properties . . . . . 1050
Output properties . . . . . 1050
Exceptions . . . . . 1050
Query for Multiple Rows as XML operation . . . . . 1051
Input properties . . . . . 1051
Output properties . . . . . 1051
Exceptions . . . . . 1051
Query Single Row operation . . . . . 1051
Input properties . . . . . 1052
Output properties . . . . . 1052
Exceptions . . . . . 1053
About Callable Statement Info Editor . . . . . 1053
Include XPath expressions . . . . . 1054
Define parameters . . . . . 1054
Test . . . . . 1055
About SQL Statement Info Editor . . . . . 1055
Include XPath expressions . . . . . 1055
Include parameters . . . . . 1056
Test . . . . . 1056
About XML Document Info Editor . . . . . 1056
Root Element Name . . . . . 1056
Repeating Element Name . . . . . 1057
Column Name Mappings . . . . . 1057
Example . . . . . 1058
JDBC exceptions . . . . . 1058
JDBCConnectionException . . . . . 1058
JDBCIllegalParameterException . . . . . 1058
SQLException . . . . . 1058
JNDIContextUnavailableException . . . . . 1058
LCCPLM . . . . . 1059
getDataAndMetaData operation . . . . . 1059
Input properties . . . . . 1059
Output properties . . . . . 1059
Exceptions . . . . . 1061
saveCommentsFromDocument operation . . . . . 1061

Input properties . . . . . 1061
Exceptions . . . . . 1061
saveCommentsFromString operation . . . . . 1061
Input properties . . . . . 1061
Exceptions . . . . . 1061
storePDF . . . . . 1062
Input properties . . . . . 1062
Output properties . . . . . 1062
Exceptions . . . . . 1062
LCCPLM exceptions . . . . . 1062
LccplmException . . . . . 1062
LDAP . . . . . 1063
Directory structure . . . . . 1063
Distinguished name . . . . . 1063
Base DN . . . . . 1065
LDAP service configuration . . . . . 1065
Initial Context Factory . . . . . 1065
Provider URL . . . . . 1065
User Name . . . . . 1066
Password . . . . . 1066
Other Properties . . . . . 1066
LDAP Query operation . . . . . 1066
LDAP Query Options properties . . . . . 1066
Connection Settings properties . . . . . 1067
Exceptions . . . . . 1067
LDAP Query to XML operation . . . . . 1068
LDAP and XML Options properties . . . . . 1068
Connection Settings properties . . . . . 1068
Output XML Document properties . . . . . 1069
Exceptions . . . . . 1069
LDAP exceptions . . . . . 1069
LDAPConnectionException . . . . . 1069
LDAPQueryException . . . . . 1069
LDAPDOMException . . . . . 1070
LDAP Query Options Editor . . . . . 1070
Query tab . . . . . 1070
Output (LDAP Query) tab . . . . . 1071
Output (LDAP Query To XML) tab . . . . . 1072
Output Settings (LDAP Query) tab . . . . . 1073
Test tab . . . . . 1074
Search Query Builder . . . . . 1074
Using the editing area . . . . . 1075
Referencing process data . . . . . 1075
Specifying attribute conditions . . . . . 1075
Inserting syntactical elements . . . . . 1076
Search filter syntax . . . . . 1076

Substrings and any values . . . . . 1077
Logical operators . . . . . 1077
Escape character . . . . . 1078
All directory items . . . . . 1078
Output . . . . . 1078
Retaining legacy appearance of PDF Forms in AEM Forms . . 1079
generatePDFOutput operation . . . . . 1079
Input properties . . . . . 1080
Template Options properties . . . . . 1080
PDF Output Options properties . . . . . 1081
General Render Options properties . . . . . 1085
PDF Specific Render Options properties . . . . . 1086
PDFA Specific Render Options properties . . . . . 1088
Output properties . . . . . 1089
Additional Output properties . . . . . 1089
Exceptions . . . . . 1089
generatePrintedOutput operation . . . . . 1089
Input properties . . . . . 1090
Template Options properties . . . . . 1091
Output Options properties . . . . . 1093
Batch Options properties . . . . . 1094
Rule Options properties . . . . . 1094
Destination Options properties . . . . . 1095
Printer Options properties . . . . . 1096
MetaData Options properties . . . . . 1097
Output properties . . . . . 1099
Additional Output properties . . . . . 1099
Exceptions . . . . . 1099
sendToPrinter operation . . . . . 1099
Input properties . . . . . 1100
Exceptions . . . . . 1101
transformPDF operation . . . . . 1101
Input properties . . . . . 1102
Output properties . . . . . 1103
Exceptions . . . . . 1103
Output exceptions . . . . . 1103
OutputException . . . . . 1103
Configuring Output operations to handle all failures . . . . . 1104
Handle exceptions that are indicated in the Status Document: 1105
Handle exceptions that throw exception events: . . . . . 1105
PDF Utilities . . . . . 1105
Clone PDF operation . . . . . 1106
Input properties . . . . . 1106
Output properties . . . . . 1106
Exceptions . . . . . 1106
Convert PDF to XDP operation . . . . . 1106

Input properties . . . . . 1106
Output properties . . . . . 1107
Exceptions . . . . . 1107
Convert XDP to PDF operation . . . . . 1107
Input properties . . . . . 1107
Output properties . . . . . 1107
Exceptions . . . . . 1107
Get PDF Properties operation . . . . . 1108
Input properties . . . . . 1108
Output properties . . . . . 1108
Exceptions . . . . . 1108
Get PDF Save Mode operation . . . . . 1109
Input properties . . . . . 1109
Output properties . . . . . 1109
Exceptions . . . . . 1109
Multiple Clones of PDF operation . . . . . 1109
Input properties . . . . . 1109
Output properties . . . . . 1110
Exceptions . . . . . 1110
Set PDF Save Mode operation . . . . . 1110
Input properties . . . . . 1110
Output properties . . . . . 1111
Exceptions . . . . . 1111
Redact PDF operation . . . . . 1111
Input properties . . . . . 1111
Output properties . . . . . 1112
PDF Utilities exceptions . . . . . 1112
PDFUtilityException . . . . . 1112
Process Engine Connector for IBM FileNet . . . . . 1112
Dispatch Workflow Step operation . . . . . 1112
Login Settings properties . . . . . 1113
FileNet Workflow Step Information properties . . . . . 1113
Exceptions . . . . . 1113
Retrieve Workflow Step Parameters operation . . . . . 1113
Login Settings properties . . . . . 1113
FileNet Workflow Step Information properties . . . . . 1113
FileNet Workflow Step Parameters properties . . . . . 1114
Result properties . . . . . 1115
Exceptions . . . . . 1115
Set Workflow Step Parameters operation . . . . . 1115
Login Settings properties . . . . . 1115
FileNet Workflow Step Information properties . . . . . 1115
FileNet Workflow Step Parameters properties . . . . . 1116
Exceptions . . . . . 1116
Process Engine Connector for IBM FileNet exceptions . . . . 1117
RepositoryException . . . . . 1117

Print PDF Package . . . . . 1117
Invoke operation . . . . . 1117
Input properties . . . . . 1117
Output properties . . . . . 1117
Invocation Policy properties . . . . . 1117
Acrobat Reader DC extensions . . . . . 1118
Apply Usage Rights operation . . . . . 1118
Input properties . . . . . 1119
Output properties . . . . . 1120
Exceptions . . . . . 1120
Get Document Usage Rights operation . . . . . 1120
Input properties . . . . . 1120
Output properties . . . . . 1121
Exceptions . . . . . 1121
Get Credential Usage Rights operation . . . . . 1121
Input properties . . . . . 1121
Output properties . . . . . 1121
Exceptions . . . . . 1121
Remove Usage Rights operation . . . . . 1121
Input properties . . . . . 1122
Output properties . . . . . 1122
Exceptions . . . . . 1122
Acrobat Reader DC extensions exceptions . . . . . 1122
ReaderExtensionsException . . . . . 1122
Render Form Guide . . . . . 1122
invoke operation . . . . . 1123
Input properties . . . . . 1123
Output properties . . . . . 1123
Invocation Policy properties . . . . . 1123
Render HTML Form . . . . . 1124
invoke operation . . . . . 1124
Input properties . . . . . 1124
Output properties . . . . . 1125
Invocation Policy properties . . . . . 1125
Render PDF Form . . . . . 1125
invoke operation . . . . . 1126
Input properties . . . . . 1126
Output properties . . . . . 1126
Invocation Policy properties . . . . . 1127
Repository . . . . . 1127
Read Resource Content operation . . . . . 1127
Input properties . . . . . 1127
Output properties . . . . . 1127
Exceptions . . . . . 1128
Repository Exceptions . . . . . 1128
RepositoryException . . . . . 1128

Document security . . . . . 1128
Apply policy operation (deprecated) . . . . . 1128
Input properties . . . . . 1128
Output properties . . . . . 1129
Exceptions . . . . . 1130
Create policy from existing policy operation . . . . . 1130
Policy properties . . . . . 1130
Validity properties . . . . . 1131
Principals properties . . . . . 1131
Permissions properties . . . . . 1132
Watermark properties . . . . . 1132
Exceptions . . . . . 1132
Create policy from template operation (deprecated) . . . . . 1132
Input properties . . . . . 1132
Exceptions . . . . . 1134
Get all policy names within a policy set operation . . . . . 1134
Input properties . . . . . 1134
Output properties . . . . . 1134
Exceptions . . . . . 1135
Get all policy set names operation . . . . . 1135
Output properties . . . . . 1135
Exceptions . . . . . 1135
Get license identifier operation . . . . . 1135
Input properties . . . . . 1135
Output properties . . . . . 1136
Exceptions . . . . . 1136
Get policy by policy identifier operation . . . . . 1136
Input properties . . . . . 1136
Output properties . . . . . 1136
Exceptions . . . . . 1136
Inspect Protected Document operation . . . . . 1136
Input properties . . . . . 1137
Output properties . . . . . 1137
Exceptions . . . . . 1137
Protect Document operation . . . . . 1137
Input properties . . . . . 1137
Output properties . . . . . 1138
Exceptions . . . . . 1139
Remove policy security operation . . . . . 1139
Input properties . . . . . 1139
Output properties . . . . . 1140
Exceptions . . . . . 1140
Revoke license operation . . . . . 1140
Input properties . . . . . 1140
Exceptions . . . . . 1141
Switch policy operation . . . . . 1141

Input properties . . . . . 1141
Exceptions . . . . . 1141
Unlock policy-protected PDF operation . . . . . 1142
Input properties . . . . . 1142
Output properties . . . . . 1142
Exceptions . . . . . 1142
Unrevoke license operation . . . . . 1142
Input properties . . . . . 1143
Exceptions . . . . . 1143
Update policy operation . . . . . 1143
Input properties . . . . . 1143
Exceptions . . . . . 1144
Document security exceptions . . . . . 1144
SDKException . . . . . 1144
Select users and groups . . . . . 1144
Add users and groups . . . . . 1144
Select users . . . . . 1145
Select groups . . . . . 1146
Set Value . . . . . 1146
execute operation . . . . . 1147
Mapping properties . . . . . 1147
Signature . . . . . 1147
Digital signatures and proxy servers . . . . . 1148
Signature service configuration . . . . . 1148
Transport Options . . . . . 1150
Path Validation Options . . . . . 1151
Timestamp Provider Options . . . . . 1152
Certificate Revocation List Options . . . . . 1153
Online Certificate Status Protocol Options . . . . . 1154
Error Handling Options for Debugging . . . . . 1156
Add Invisible Signature Field operation . . . . . 1156
Common properties . . . . . 1157
Advanced properties . . . . . 1157
Output properties . . . . . 1161
Exceptions . . . . . 1162
Add Signature Validation Information operation . . . . . 1162
Common properties . . . . . 1162
Signature Validation Options properties . . . . . 1162
Output PDF properties . . . . . 1170
Exceptions . . . . . 1170
Add Visible Signature Field operation . . . . . 1170
Common properties . . . . . 1171
Advanced properties . . . . . 1171
Output properties . . . . . 1175
Exceptions . . . . . 1175
Certify PDF operation . . . . . 1175

Common properties . . . . . 1176
Appearance properties . . . . . 1179
Advanced properties . . . . . 1181
Output properties . . . . . 1187
Exceptions . . . . . 1188
Clear Signature Field operation . . . . . 1188
Input properties . . . . . 1188
Output properties . . . . . 1189
Exceptions . . . . . 1189
Get Certifying Signature Field operation . . . . . 1189
Input properties . . . . . 1189
Output properties . . . . . 1189
Exceptions . . . . . 1189
Get Legal Attestation operation . . . . . 1190
Input properties . . . . . 1190
Output properties . . . . . 1190
Exceptions . . . . . 1190
Get Signature operation . . . . . 1190
Input properties . . . . . 1191
Output properties . . . . . 1191
Exceptions . . . . . 1192
Get Signature Field List operation . . . . . 1192
Input properties . . . . . 1192
Output properties . . . . . 1192
Exceptions . . . . . 1192
Get Signed Version operation . . . . . 1192
Input properties . . . . . 1193
Output properties . . . . . 1193
Exceptions . . . . . 1193
Modify Signature Field operation . . . . . 1194
Input properties . . . . . 1194
Outputproperties . . . . . 1199
Exceptions . . . . . 1199
Remove Signature Field operation . . . . . 1199
Input properties . . . . . 1199
Output properties . . . . . 1200
Exceptions . . . . . 1200
Sign Signature Field operation . . . . . 1200
Commonproperties . . . . . 1200
Appearance properties . . . . . 1202
Advanced properties . . . . . 1205
Output properties . . . . . 1210
Exceptions . . . . . 1211
Verify PDF Document operation . . . . . 1211
Common properties . . . . . 1211
PKI Options properties . . . . . 1211

SPI Options properties . . . . . 1218
Output properties . . . . . 1219
Exceptions . . . . . 1219
Verify PDF Signature operation . . . . . 1219
Common properties . . . . . 1219
PKI Options properties . . . . . 1220
SPI Options properties . . . . . 1227
Output properties . . . . . 1227
Exceptions . . . . . 1228
Verify PDF Signature operation (deprecated) . . . . . 1228
Common properties . . . . . 1228
Advanced properties . . . . . 1229
Output properties . . . . . 1237
Exceptions . . . . . 1237
Verify XML Signature operation . . . . . 1237
Common properties . . . . . 1238
Advanced properties . . . . . 1239
Output properties . . . . . 1245
Exceptions . . . . . 1245
Signature exceptions . . . . . 1245
CredentialLoginException . . . . . 1245
DuplicateSignatureFieldException . . . . . 1245
FIPSComplianceException . . . . . 1245
InvalidArgumentException . . . . . 1245
MissingSignatureFieldException . . . . . 1245
NoCertifyingSignatureException . . . . . 1245
PDFOperationException . . . . . 1245
PermissionsException . . . . . 1246
SeedValueValidationException . . . . . 1246
SignatureFieldNotSignedException . . . . . 1246
SignatureFieldSignedException . . . . . 1246
SignatureVerifyException . . . . . 1246
SigningException . . . . . 1246
SPI Properties . . . . . 1246
Add Subject DN . . . . . 1247
Stall . . . . . 1249
execute operation . . . . . 1249
Input properties . . . . . 1249
Submit Form Guide . . . . . 1250
invoke . . . . . 1250
Input . . . . . 1250
Output . . . . . 1251
Invocation Policy . . . . . 1251
Submit HTML Form . . . . . 1251
invoke operation . . . . . 1251
Input properties . . . . . 1252

Output properties . . . . . 1252
Invocation Policy properties . . . . . 1253
Submit PDF Form . . . . . 1253
invoke operation . . . . . 1253
Input properties . . . . . 1253
Output properties . . . . . 1254
Invocation Policy properties . . . . . 1254
User 2.0 . . . . . 1255
Assign Multiple Tasks operation . . . . . 1255
Participants properties . . . . . 1255
Task Instructions properties . . . . . 1255
User Actions properties . . . . . 1256
Presentation & Data properties . . . . . 1256
Output properties . . . . . 1258
Workspace User Interface properties . . . . . 1258
Task Access Control List (ACL) properties . . . . . 1259
Attachments properties . . . . . 1260
Priority properties . . . . . 1260
Reminders properties . . . . . 1260
Deadline properties . . . . . 1261
Custom Email Templates properties . . . . . 1262
Exceptions . . . . . 1262
Assign Task operation . . . . . 1263
Initial User Selection properties . . . . . 1263
Task Instructions properties . . . . . 1264
User Actions properties . . . . . 1264
Presentation & Data properties . . . . . 1265
Output properties . . . . . 1266
Workspace User Interface properties . . . . . 1267
Task Access Control List (ACL) properties . . . . . 1268
Reassignment Restrictions properties . . . . . 1269
Attachments properties . . . . . 1270
Priority properties . . . . . 1270
Reminders properties . . . . . 1270
Escalation properties . . . . . 1271
Deadline properties . . . . . 1273
Custom Email Templates properties . . . . . 1273
Exceptions . . . . . 1274
Assigning tasks across development and production environments 1274
User exceptions . . . . . 1274
InvalidPrincipal . . . . . 1274
About Select User . . . . . 1275
About Select Group . . . . . 1277
About Email Template Editor . . . . . 1278
User Lookup . . . . . 1280
Find Group operation . . . . . 1280

Filter properties . . . . . 1280
Result properties . . . . . 1281
Exceptions . . . . . 1281
Find Groups operation . . . . . 1281
Filter properties . . . . . 1281
Result properties . . . . . 1282
Exceptions . . . . . 1282
Find User operation . . . . . 1283
Filter properties . . . . . 1283
Result properties . . . . . 1284
Exceptions . . . . . 1284
Find Users operation . . . . . 1284
Filter properties . . . . . 1284
Result properties . . . . . 1285
Exceptions . . . . . 1286
findGroupMembers operation . . . . . 1286
Input properties . . . . . 1287
Output properties . . . . . 1287
Exceptions . . . . . 1287
User Lookup Exceptions . . . . . 1288
UMException . . . . . 1288
Variable Logger . . . . . 1288
log operation . . . . . 1288
System Logging properties . . . . . 1288
Directory Logging properties . . . . . 1289
Exceptions . . . . . 1290
Variable Logger exceptions . . . . . 1290
Java.io.IOException . . . . . 1290
VariableLoggerConfigurationException . . . . . 1290
VariableLoggerExecutionException . . . . . 1290
Wait Point . . . . . 1290
businessCalendarWait operation . . . . . 1290
Input properties . . . . . 1290
scheduleWait operation . . . . . 1291
Input properties . . . . . 1291
Web Service . . . . . 1291
Web Service service configuration . . . . . 1292
Invoke Web Service operation . . . . . 1292
Web Service Options properties . . . . . 1292
Web Service Response properties . . . . . 1292
Exceptions . . . . . 1293
Web Service exceptions . . . . . 1293
About Web Service Settings . . . . . 1293
Settings tab . . . . . 1293
HTTP Settings tab . . . . . 1295
Request tab . . . . . 1295

Attachment tab . . . . . 1296
Test tab . . . . . 1297
About SOAP invocation messages . . . . . 1297
Namespaces . . . . . 1298
Example . . . . . 1298
Invoking services in AEM Forms using Web Services . . . . . 1299
Invoking services in AEM Forms with document input and output values
1299
Configuring request messages . . . . . 1300
Configuring web service response messages . . . . . 1303
Using Table data types for SAP web services . . . . . 1303
XMP Utilities . . . . . 1304
Export Metadata operation . . . . . 1305
Input properties . . . . . 1305
Output properties . . . . . 1305
Exceptions . . . . . 1305
Import Metadata operation . . . . . 1305
Input properties . . . . . 1306
Output properties . . . . . 1306
Exceptions . . . . . 1306
Export XMP operation . . . . . 1306
Input properties . . . . . 1306
Output properties . . . . . 1307
Exceptions . . . . . 1307
Import XMP operation . . . . . 1307
Input properties . . . . . 1307
Output properties . . . . . 1307
Exceptions . . . . . 1307
XMP Utilities exceptions . . . . . 1308
XMPUtilityException . . . . . 1308
XSLT Transformation . . . . . 1308
XSLT Transformation service configuration . . . . . 1308
Factory Name . . . . . 1308
Transform operation . . . . . 1308
Factory Options properties . . . . . 1308
XML Source properties . . . . . 1309
XSLT Source properties . . . . . 1309
Transformation Result properties . . . . . 1309
Exceptions . . . . . 1309
Transform using XSLT(URL) operation . . . . . 1309
Factory Options properties . . . . . 1309
XML Source properties . . . . . 1310
XSLT Source properties . . . . . 1310
Transformation Result properties . . . . . 1310
Exceptions . . . . . 1310
XSLT Transformation exceptions . . . . . 1310

Java.lang.IllegalAccessException . . . . . 1310
Java.net.MalformedURLException . . . . . 1311
Java.io.IOException . . . . . 1311
Javax.xml.transform.TransformerException . . . . . 1311
Java.net.URISyntaxException . . . . . 1311
Javax.xml.parsers.ParserConfigurationException . . . . . 1311
Org.xml.sax.SAXException . . . . . 1311
Java.lang.ClassNotFoundException . . . . . 1311
Java.lang.InstantiationException . . . . . 1311
Java.lang.IllegalAccessException . . . . . 1311
About XSLT Source Text Input and Testing . . . . . 1311
XSLT . . . . . 1311
Test XML . . . . . 1311
Test Result . . . . . 1312

**Index . . . . . i**

# 1. About AEM Forms Workbench

Workbench is an integrated development environment (IDE) that developers use to create and manage AEM Forms applications and assets.

## 1.1. About the user interface

Workbench has several common features, such as the menu bar and toolbar. It also displays a collection of windows or panels, called *views* or *editors*. The collection of views and editors is called a *perspective*.

The *menu bar* provides access to a set of commands. The *toolbar* provides quick access to commands that you can frequently use to create processes and forms.

*Views* contain data entry fields, file hierarchy lists, buttons, and other such tools that you use to specify details about processes and forms. Many views offer additional commands in context menus, which you display by right-clicking in the view.

### Perspectives

A *perspective* is a group of views and editors for accomplishing a specific type of task. Workbench includes several perspectives that are optimized for developing specific application assets:

- Process Design (See Openingthe Process Design perspective.)
- Form Design (See Openingthe FormDesign perspective.)
- (Deprecated) Document Builder (See GettingStarted with (Deprecated) Document Builder)
- Guide Design (See GuideDesign perspective)
- Data Model (See Usingdata models with processes.)

  *Workbench also provides the AEM Forms Runtime View perspective. This perspective is useful for seeing all of the assets that are active on the AEM forms server. This perspective is useful when your environment includes assets created using AEM Forms and resources, processes, and event types that were created using previous versions. (See Leveraginglegacy solutions in AEM Forms.)*

  *You can switch to another perspective by selecting Window > Open Perspective or by clicking one of the perspective buttons in the toolbar. You can switch between perspectives freely, but only one is displayed at a time.*

  *The perspective that was last active is reactivated the next time you start Workbench.*

  *You can customize a perspective to suit your personal preferences by selecting, placing, and sizing the editors and views. Your personal preferences can change and are dependent on what you want to accomplish for your development tasks.*

### Views

A *view*, which is a tab within a window, contains a set of elements such as a navigation tree or fields for setting property values. Views include menus and toolbars. You can open and close views and dock them in other locations.

A perspective has a defined set of views. The views in a perspective support the tasks you perform in that perspective. For example, when you are creating processes, the views displayed in the Process Design perspective relate to drawing and configuring process diagrams.

### Editors

An *editor* allows you to create and edit objects or files of various types, such as process diagrams and forms. The editor opens automatically when required. For example, if you open a form, the associated editor opens.

More than one editor can be open at once. Tabs in the editor area indicate the names of the objects or files that are currently open for editing. An asterisk (*) beside the name indicates that the file or object has unsaved changes.

As with views, you can open, close, and dock editors to suit your preferences.

### Preferences

Preferences are available for personalizing the behavior of Workbench when you open assets. Select or deselect the following options and click Apply.

**Automatically Open Associated Perspectives When Editing Assets:**

Select to automatically switch to the perspective that is associated with the type of asset that you open. For example, when you open a process, the Process perspective is opened.

**Perform Asset(s) Checkout Automatically:**

Select to automatically check out assets when you open them.

RELATED LINKS:

CustomizingPerspectives

## 1.2. Related software

As you learn about Workbench, be aware of the following related AEM Forms modules:

| Related software | Description |
|---|---|
| AEM forms server | The repository for process diagrams, forms, and other resources used in the business process. The server can be either a single server or a server cluster.<br>To work in Workbench, it is necessary to log in to a AEM Forms server. |
| User Management pages of the AEM Forms Administration Console | Use for creating user accounts and user groups, and configuring their security permissions. See Settingup and organizing users in Administration Help |
| Applications and Services pages of the Administration Console | Use for importing and configuring archive files that you create using Workbench. For more info see Applications and ServicesAdministrationHelp. |
| forms workflow pages of administration console | Use for configuring server settings, and administering process instances at run time. For example, you can get information about stalled operations and task assignments. See forms workflow administrationhelp. |
| Workspace | A user-facing interactive portal from which users can initiate processes and work with forms in a process. As a developer, you can customize the appearance of Workspace, extend its components, and utilize its exposed components to customize it. |

## 1.3. Adobe Community Help Client (CHC)

The CHC is an AIR-based application that replaces the Eclipse help engine for Workbench and is the platform for the next generation of Adobe help delivery. CHC features include:

- Always online

  *If you have a network connection, the CHC accesses content from the web. This ensures that you access the most up-to-date-material. It can also work in local mode if there is no Internet connection.*
- Search-centric

  *Use Community Help search, adobe.com search, or local search. Community Help search aggregates resources, including those from 3rd party sites. adobe.com search includes refinements to narrow your scope.*
- In-context navigation

  *Provides a dynamically generated set of related links for key pages.*

# 2. What's new in Workbench

Workbench introduces several new features that simplify the development of AEM Forms applications.

For information about continuing the development of legacy processes, resources, and event types in AEM Forms, see Leveraginglegacy solutions in AEM Forms.

## 2.1. New for Workbench

Workbench has the following process designer enhancements to enable effective process design with increased efficiency.

### Process designer enhancements

Take advantage of the following enhancements to enable effective process design.

#### Looping in processes

In earlier versions of Workbench, circular routes had to be added manually to implement looping. The Iteration Wizard in Workbench allows you to automatically generate loops. Variables are generated to track the iteration of the loop. See Loopingin Processes for more information.

#### Grouping in processes

Group logically and functionally coherent activities on your process diagram using the grouping feature. See Groupingin Processes for more information.

### Automating legacy solutions upgrade

Workbench is equipped with tools to automate upgrading your legacy solutions, archive files, and artifacts (processes, forms, form fragments, images, ddx files, etc.).

#### Archive Migration tool

Workbench allows you to migrate your AEM Forms archive files to the application model. The tool was available as a plug-in for earlier versions, but is a standard feature of Workbench. See MigratingLCAs to AEM Forms for more information.

#### Upgrade legacy processes and artifacts

The Upgrade Legacy Artifacts feature allows you to upgrade your legacy artifacts and bundle them in to AEM Forms applications. You can select a process and all the artifacts that it depends on to be upgraded and bundled in to a single application. See Upgradinglegacy artifacts for more information.

## Dependency detection

You can track asset dependencies using the Dependency Detection feature. Workbench notifies if you attempt changing the state or properties of an asset that might affect dependencies and hence, the functioning of dependent assets. See Managingasset dependenciesfor more information.

### Visualizing dependencies

You can also, at any point in time, view the dependencies of an asset using the Dependency Visualization feature. See Visualizingdependencies for more information.

### Asset renaming support

Assets can now be renamed without breaking references and affecting dependencies. Workbench allows you to dynamically update references when renaming is attempted. See Managingasset dependencies for more information.

## Timer enhancements to Event Behavior Configuration

You can now specify literal time values when configuring Event Throws and Event Receives. The lowest available granularity is Seconds, which facilitates enhanced exactness when scheduling. See Controlling-process flow using events for more information.

## Workbench offline notifications

Enhanced warning messages to help track and gracefully handle server time outs.

## Performance enhancements

- Lesser refresh time with Drag and Drop operation of a file set to an application
- Efficient and faster Check in and Check out operations
- Improved time efficiency of LCA creation
- Improved time efficiency in getting applications from server, synchronizing, deploying, undeploying, and deleting applications.
- Faster access to remote resources

## Common variables

AEM Forms allows you to define a set of variables that can be made common to all long-lived processes. This enhances searchability in Workspace where you can search all processes that use a common variable. See CreatingCommon Variables for more information.

## Persisting XML form data with Data Models

You can persist form and guide data that are based on data models with AEM Forms Data Models. Model-based forms and Guides require XML data that is structured according to the same data model. See Usingdata models with form and Guide data for information.

## Integrating Data Models with Web Services

You can now integrate Web Services with AEM Forms Data Models. Integration is supported for WSDL, SOAP - based, REST - based, and HTTP - based web services. See Integratingwith Web Services for more information.

## Integrating Data Models with Data Services

You can now integrate Data Services with AEM Forms Data Models. See Integratingwith Data Services for more information.

## Building expressions with referred XSDs

Workbench allows you to build expressions using nodes of XML variables that conform to XSD that reference other XSD files. See CreatingXPath expressions for more information.

# 3. Leveraging legacy solutions in AEM forms

Workbench enables you to continue using legacy solutions. After your AEM Forms environment has been upgraded to AEM Forms, you can access legacy solution components for maintenance and further development.

For an overview of the changes that have been implemented in Workbench, see What'snew in Workbench.

## 3.1. About the upgraded environment

Application Model (in practice with Workbench 9.0 and latter versions) allows multiple process developers to share assets across applications. It also allows securitization of confidential assets and applications shared on a AEM Forms Server for restricted usage. Legacy artifacts (AEM Forms assets) that exist as Resources (processes, forms, form fragments, images, ddx files, etc.)in the upgraded environment can be extracted and bundled in to a single container called Application. Applications can be versioned. This means that the resources do not have versions of their own and they inherit the versions of their application containers. Application version numbers are uneditable, meaning the version numbers do not change even if they are migrated across servers or development environments.

When the AEM Forms environment is upgraded to AEM Forms, the properties of legacy processes, resources, and events are unchanged. For example, if a process was locked before the upgrade in Workspace, the process appears with a lock icon on it after the upgrade. Consider unlocking all processes, resources and event types before you import them to AEM Forms. In Workbench, you cannot lock/unlock processes.

The following summary describes what occurs to an upgraded legacy asset:

**Processes and process versions:**

Remain in the Processes view. Lock, activation, and recording properties remain unchanged.

**Resources:**

Remain in the Resources view. Resource locks remain unchanged.

**Event types:**

Remain in the Events view. Locks and activation properties remain unchanged.

After you upgrade, check-in and deploy the process, process version, resource, or event to remove the lock property.

The run-time instances of legacy processes, resources, and event types continue to execute in the AEM Forms environment. References between processes, resources, and events are persisted:

- Processes continue to execute correctly.
- Forms continue to reference form fragments and other resources correctly.

Additionally, service endpoints remain unchanged. Processes can be invoked using the same endpoints that existed in the AEM Forms environment.

In Workbench, interaction with items in the Processes, Resources, and Events view is limited. Many features and commands have moved from these views to the new Applications view. For example, you cannot open resources for editing or viewing from the Resources view.

### Review the AEM Forms run-time environment:

1) Open the following perspectives and views:
   – AEM Forms Runtime View perspective (includes the Applications, Resources, Services, and Events views)
   – Processes view

## 3.2. Strategies for leveraging legacy solutions

Determine whether you will continue to use legacy solutions as-is, or use them as the starting point for developing AEM Forms assets. Consider following factors when you make a decision:

- Your satisfaction with current solutions.
- The resources required to maintain current solutions.
- The resources required to upgrade legacy solutions so that you can use new AEM forms features.
- The benefits of using new AEM Forms features. (See What'snew in Workbench.*)*

### Continued execution

After upgrading the environment to AEM Forms, leave legacy items unchanged in the Processes, Resources, and Events views. Processes, resources, and events that were used in the AEM Forms 8.x environment are fully supported in AEM Forms. Your solutions continue to function as they did in the legacy environment. This strategy is suitable for mature solutions that require no maintenance.

### Maintenance of legacy items

Change legacy processes, resources, and events using Workbench while preserving their link to legacy runtime instances. After importing into AEM forms applications, your processes, resources, and custom events continue to execute as AEM Forms 8.x solutions. However, when you preserve the link to legacy runtime instances, you cannot take advantage of new AEM Forms features.

For example, edit a process to fix a bug. The changes affect existing process instances and new instances as the process is invoked. After you implement the changes, you can deploy the application. To propagate the changes to another environment, create a AEM Forms 8.x archive or update an existing AEM Forms 8.x archive.

This strategy is suitable for solutions that completely satisfy your business needs, but require occasional maintenance.

**To maintain legacy items:**

1) Import legacy items to an application. (See Importingresources, processes, and events to applications.)
2) Edit the items and deploy them or add to AEM Forms 8.x archives for deployment to production. (See Maintaininglegacy run-time instances.)

### Progressive development

Break the link between legacy processes, resources, and events and their legacy runtime instances and create full-fledged AEM Forms application assets:

- Take advantage of new features.
- Develop assets independent of their legacy runtime counterparts.
- Legacy processes, resources, and events remain intact.

**To create AEM Forms assets from legacy items:**

1) Import legacy items to an application. (See Importingresources, processes, and events to applications.)
2) Break the link to the legacy runtime instance. (See Upgradinglegacy solutions to AEM Forms.)

## 3.3. Importing resources, processes, and events to applications

Import legacy processes, resources, and events to add them to applications in the Applications view. For each legacy item, a corresponding asset is created in a target application.

The following rules apply when importing legacy items to applications:

- You can import only one version of a process to an application version. If you wish to import multiple versions of a process, see Importingmultiple versions of a process for more information.
- You can import a process, resource, or event to multiple applications to create multiple copies.
- Applications can include either legacy items or assets created using AEM Forms. Do not place both legacy items and AEM Forms assets in the same application.
- In an application, process names must be unique. If the process you are importing has the same name of an existing process, even if their deployment IDs are different, the import operation fails.
- In an application, Deployment IDs of processes must be unique. If the process you are importing has a different name but the same Deployment ID of an existing process, the import operation fails.

Application names and folder names have the following restrictions:

- Names can have a maximum length of 40 characters.
- The total number of characters in the path of an asset cannot exceed 256.
- Names cannot include control characters (characters that have an ASCII value that is less than 32).
- Names cannot include any of the following characters: "/", "\", "+", "$", "?", "*", ":", "|", "<", ">", ",", and the tab key.

If your legacy resources use a folder structure that conflicts with these restrictions, modify the folder structure before importing. Not adhering to these restrictions can cause resources to be inaccessible.

You can import legacy items from the run-time environment, the file system, and AEM Forms 8.x archives.

**TIP:** Check in imported items to save them on the AEM Forms Server. (See Developing applicationsin a multi-developer environment.*)*

## Importing multiple versions of a process

Workbench does not allow you to import multiple versions of a legacy process simultaneously. If you wish to import multiple versions of a process:

1) Create a new application to import the process.
2) Import any one version of the process at a time.
3) Reset its deployment ID. When you reset the deployment ID, the legacy process is converted to a AEM Forms compatible process. See Breakingthe link to existing run-time instances for more information.
4) Repeat the steps 1 to 3 for each version of the process you wish to import.

## Importing from the run-time environment

Import legacy items from the run-time environment to create representations of them in the Applications view. The resultant application assets are directly linked to the instances in the run-time environment.

Typically, you import legacy items from the run-time environment to edit them. For example, to edit a process version that exists in the run-time environment, import it to an application and then open it. (See Maintaininglegacy run-time instances.)

After importing, the locations in the run-time environment do not change. Consequently, references between imported items are persisted regardless of which application they belong to. Therefore, legacy applications continue to execute successfully.

**NOTE:** Create applications before importing legacy items. (See Addingand removing applications.)

**Import legacy items from the run-time environment:**

1) Click File > Import.
2) Under AEM Forms Runtime Content, select Event, Process, or Resource, depending on the type of item you are importing, and then click Next.
3) Select the event types, resources, or process versions to import:
   – **Events:**Select one or more event types. To select multiple event types, expand the event categories and Ctrl+click.
   – **Processes:**Select one or more process versions. To select multiple process versions, expand the process groups and processes and then Ctrl+click.

– **Resources:**Select resources or resource folders. If you select a folder, the entire contents, including subfolders and resources, are imported. Ctrl+click to select multiple folders or resources.

**NOTE:** Do not import or upgrade pre populated system processes that are provided with Workbench. For example, forms workflow Email (system).

4) In the Enter Or Select Import Location box, specify the target location for the selected items, and then click Finish.

## Importing from the file system

Import AEM forms 8.x items from the file system when you want to use them in AEM Forms applications. For example, import processes that you exported to XML files from the AEM Forms 8.x environment.

You can import individual files or entire folders. You can also duplicate the file structure on the file system in the application.

After you import files with long file structures, verify that the path does not exceed the 256-character limit. When the imported file has a resulting path in the application that exceeds the 256-character limit, the item is inaccessible from Workbench.

**NOTE:** Create applications before importing legacy items. (See Addingand removing applications.)

**Import from the file system:**

1) Click File > Import.
2) Click General > File System and then click Next.
3) Browse for the folder that contains the file to import.

   **NOTE:** If you import a file that is in a subfolder of the folder you select, the subfolder is also created in the application.
4) Select entire folders or individual files to import.

   **TIP:** You can select files from multiple folders.
5) Click the Browse button next to the Into Folder box and select the application folder to contain the files.

   **TIP:** To see only the file types that you are interested in, click Filter Types and select those file types.
6) To replace assets in the application that have the same name and location as the items that you are importing, select Overwrite Existing Resources Without Warning. If you do not select this option, you decide whether to replace same-named files when they are being imported.
7) Duplicate the folder structure of the file you are importing, including all folders to the root of the file system. Select Create Complete Folder Structure, and then click Finish.

## Exporting files referenced by literal values

Workbench does not support literal values locally referencing files from the file system; outgoing file references can only be made to files stored in the AEM Forms Server. However, if your legacy AEM Forms

8.x process uses literal values that locally reference files from the file system, Workbench allows you to export these files to the server after upgrading your process:

1) Use the Upgrade ES Artifacts tool to upgrade the AEM Forms 8.x process. For more information, see UpgradingLegacy Artifacts.
2) Check out and open the process in the process designer.
3) In the Process Properties View, navigate to the literal value and click the button.
4) Specify a location on the server for the file to be exported. You could also create a new folder by clicking **New Folder**.
5) Click **Ok** to finish exporting the file.

## Importing from AEM Forms 8.x archive files

To import items from AEM Forms 8.x archive files (that were created using AEM Forms), first import the archive files into AEM Forms using administration console. After importing, the items in the AEM Forms archive file appear in the Processes, Resources, and Events views as they did in the original AEM Forms environment. For example, process versions appear in the same process group in the Processes view.

**NOTE:** If the root of the path of imported resources is Applications, the imported resources do not appear in the Resources view. For example, an AEM Forms archive file contains the resources from the Application/LoanApplication folder. When this AEM Forms archive file is imported, the contents do not appear in the Resources view.

Create applications before importing legacy items. (See Addingand removing applications.)

### Legacy LCA files that include service configurations

In AEM Forms, when processes used services that were configured using the Components view, AEM Forms archive files included the service configuration. These service configurations cause problems when they are imported into AEM Forms environments. When you preview archive contents when importing into AEM Forms, deselect any service configurations for service components.

For example, a AEM Forms process uses the LDAP service. The connection properties for the LDAP service were configured using the Components view. When the process is packaged in a AEM Forms archive, the archive includes the service configuration. When imported, the service configuration is appears as ServiceConfiguration_LDAPService in the preview. This item should be deselected before importing into AEM Forms.

**NOTE:** Do not deselect service configurations for process services.

**Import legacy items from AEM Forms archive files:**

1) Use administration console to import the AEM Forms archive files. (See Import andmanage AEM Forms 8.x archivesin Administrative help.)
2) In Workbench, open the Processes view and refresh the view. Refreshing causes any processes that were imported from the LCA file to appear.
3) Open the Events view and refresh the view. Refreshing causes any event types that were imported from the LCA file to appear.
4) Import the items that the AEM Forms archive file contained from their run-time location. (See Importingfrom the run-time environment.)

**NOTE:** Refresh the Applications view to see the imported items.

**NOTE:** Importing a AEM Forms archive file where the application contains a folder named `components` fails. To resolve this issue, rename the folder in the AEM Forms application to another name other than components, recreate the archive file, and reimport the file to the AEM Forms Server.

# 3.4. Maintaining legacy run-time instances

After you import from the run-time environment, perform the following tasks to continue the development of your legacy processes, custom event types, and resources:

- Modify the imported items.
- Propagate changes to other environments.

## Changing existing run-time instances

By default, items that are imported to applications from the run-time environment are linked to the existing run-time instances. Consequently, changes to the imported items affect the run-time behavior of legacy solutions.

For example, a AEM Forms process is imported to an application. The form it uses is also imported. The form is checked out and opened using Adobe Designer 9. After changes are made to the form, it is checked in and the application is deployed. The changes affect all new form instances that are created when users open their tasks to participate in the process.

**Change legacy processes, events, and resources:**

1) Open the process, event type, or other asset from the Applications view, edit and save the changes. (See Editingand viewing assets.)
2) Check in the process, event type, or other asset. (See Editingand viewing assets.)
3) Deploy the application. (See Deployingapplications.)

## Updating legacy solutions in other environments

After changing legacy items in the development environment, propagate the changes to other environments, such as staging or production environments. Create or update AEM Forms 8.x archives to update legacy applications that were deployed using AEM Forms.

The Processes view and the Resources view enable you to create and update AEM Forms archive files. After you create or update the AEM Forms archive file, use administration console to import it into the other environment. (See Import and manage AEM Forms 8.x archivesin Administrative help)

When importing you import 8.x archives, matching processes, resources, and event types that exist on the target environment are not updated with the copies in the archive.

**NOTE:** AEM Forms archives are based on AEM Forms applications and are not compatible with AEM Forms 8.x archives.

The following user permissions are required for creating or updating archive files:

- To include processes in the archive, the user must have Service Read and Service Invoke permissions.
- To include resources in the archive, the user must have Repository Read and Repository Traverse permissions for the resource being exported. The user must have Repository Traverse permission for the parent folders.

**NOTE:** When locked resources are added to a AEM Forms archive, lock information is not included in the archive.

### Selecting contents

The following rules apply when selecting processes and resources:

- To make a selection, click the box beside a component that you want to select. A check mark appears in the box. To deselect a component, click the box again.
- A shaded box marks the components selected implicitly.
- Selecting a process category automatically selects each process type within the category.
- Selecting a process automatically selects the latest version of the process. You can manually select earlier versions of the process.
- Selecting a resource folder automatically selects all the files and folders within it.
- Processes and resources that are selected automatically cannot be deselected individually.
- To select several items, click the name of the first item, and then press Shift and click the name of the last item to select. You can also press Ctrl and click individual names. Then click the box beside one of the marked process names.

### Create an archive file:

1) In the Processes or Resources view, click button.
2) Select Create An Archive File and click Next.

3) (Optional) In the Archive Name box, type the name for the archive. The box is prepopulated with the words "New Archive" followed by a date and time string to ensure that it is unique. This name is displayed in administration console when you import the archive.
4) (Optional) In the Archive Description box, type a description for the archive so that others know what its contents are. A default description specifies the date and time of the archive creation and the name of the user who created the archive.
5) In the Local File System Destination box, specify the file name and location of the archive:

   **NOTE:** You do not need to specify the file name extension. It is added automatically to the file name.
6) Click Next.

   *If you clicked the Create A AEM Forms Archive From Selected Components button on the Resources view, continue with step 10.*
7) (Processes only) Select processes to include in the archive file.

   **NOTE:** Only the processes that are activated are displayed in the list.
8) (Processes only) Click Next to preview the list of processes to include in the archive file or click Finish to save the archive file immediately.

   *If you did not select any processes, clicking Next opens the Resource Selection panel.*
9) (Processes only) In the Archive Preview panel, review the list of processes that are included in the archive file.

   *The list indicates the process version and whether you explicitly selected the process or it is implicitly included. Processes are implicitly included when a selected process depends on them. A check mark in the Implicit column indicates that the process was implicitly included.*
10) Click Next to add resources to the archive file or click Finish to save the archive.
11) (Optional) Select resources to include in the archive file.

    *Resources required by the processes that are selected in the Processes Selection panel are automatically included in the archive and cannot be deselected.*
12) Click Next to preview the list of the selected components to include in the archive file or click Finish to save the archive file immediately.
13) After reviewing the contents of the archive file, click Finish to save the file.

**Update an archive file:**

1) In the Processes or Resources view, click the Create A AEM Forms Archive From Selected Components button.
2) Select Update Existing Archive File and click Next.
3) In the Existing Archive box, type the name and location of the AEM Forms archive file to update. Click the ellipsis button to browse to the location of the file.
4) Click Next to modify archive properties.
5) (Optional) The Archive Name box displays the current name of the archive. Change it as required (for example, to indicate the date the archive was updated).

   *The archive name must be unique on the system where the archive is imported. This name is displayed in administration console when you import the archive.*

6) (Optional) In the Archive Description box, type a description for the archive so that other users know what its contents are. The default description includes information about the archive creation, previous updates, and the current update.
7) (Optional) In the Local File System Destination box, change the name of the archive file to create an archive based on the one selected. To update the content of the archive, do not change the file name.
8) Click Next to add or remove processes from the archive file or click Finish to save the archive file. If you are using the same file name, a message prompts you to confirm whether you want to overwrite the existing file. Click OK.

   *When you click Finish, AEM Forms automatically updates the versions for processes and resources that were explicitly selected when the archive was originally created.*
9) (Optional) Select processes to add to the archive file or deselect processes to remove from the archive file.
10) Click Next to preview the list of processes to include in the archive file or click Finish to save the archive file immediately.

    *If you did not select any processes, clicking Next displays the Resource Selection panel.*
11) In the Archive Preview panel, review the list of processes that are included in the archive file. Click Next to add or remove resources or click Finish to save the archive.
12) (Optional) Select resources to add to the archive file or deselect resources to remove from the archive file.
13) Click Next to preview the list of the selected components to include in the archive file or select Finish to save the archive file.
14) After reviewing the contents of the archive file, click Finish to save the file.

# 3.5. Upgrading legacy solutions to AEM forms

Use legacy processes, resources, and events as starting points for developing AEM Forms solutions that take advantage of new features. After importing legacy processes, resources, and custom events into applications, make them independent from their legacy runtime instances. Then, modify the assets as required.

Make sure that you have imported all processes, resources, and custom events that are referenced. For example, if you are upgrading a process, import the process, forms, and other resources that it uses, custom events, custom render and submit processes, and subprocesses. (See Importingresources, processes, and events to applications)

RELATED LINKS:

Updatingreferences to assets

## Upgrading legacy artifacts

Workbench automates the process of upgrading legacy solutions and artifacts with the Upgrade ES Artifacts tool. You can bundle a legacy processes and its dependencies (forms, images, etc.) into a AEM

Forms application. Instead of maintaining runtime references, Workbench makes copies of legacy artifacts in the AEM Forms Server when they are selected to be included in an application. Before you attempt upgrading your process, see Aboutthe upgraded environment.

1) In Workbench, access **File** > **Upgrade ES Artifacts**.
2) On the Process Selection screen, select the processes you wish to upgrade. Note that, you can select specific versions of a process you wish to include in your application by drilling down to the appropriate process version. Use the Select All Processes option to select all processes. Click **Next**.

   **NOTE:** Do not import or upgrade pre populated system processes that are provided with Workbench. For example, forms workflow Email (system).
3) Review the summary of selected processes on the Process Preview screen. There might be processes, flagged as Implicit, listed that you may not have selected earlier. These are processes that are referenced by other dependent processes selected earlier on the Process Selection screen. To this end, they are implicitly selected to be upgraded and bundled into an application. You can choose to override these implicit selections by deselecting appropriate assets. Click **Next**.
4) On the Resource Selection screen, select the resources you wish to upgrade. Click **Next**.
5) Review the summary of your selection on the Selection Process screen. It is recommended that the number of selected resources should not exceed 400. Click **Next**.
6) Enter a name and provide a meaningful description for the application on the Enter Application Information screen. Click **Finish**.
7) Complete all the manual tasks listed in the dialog box.

***NOTE:*** *On the Process Selection screen, when folder level selections are made, latest versions of processes are selected by default. To select the previous version of a process, deselect the folder level selection.*

## Breaking the link to existing run-time instances

Break links between imported assets and their legacy run-time counterparts to change assets without affecting existing run-time instances.

For example, you want to reference a legacy form in an application in the AEM Forms environment. The new application requires you to change the form. However, the changes cause errors with a legacy process that still uses the form. So, you add the form to the application, and then break the link between the form and the run-time counterpart. Now, the changes do not affect the legacy process.

All items that are imported from the run-time environment have a value for their Deployment ID property. This property identifies the location of the item in the run-time environment. The value also persists the link between the imported asset and the run-time counterpart. To break the link, remove the value for this property.

**TIP:** You can still change the run-time instance of a legacy asset after you break the link. Import the process version, resource, or event type again from the run-time environment.

**Remove the value of Deployment ID:**

1) In the Applications view, right-click the asset and click Properties.
2) Next to the Deployment ID box, click the Reset button and then click OK.

**NOTE:** If you no longer want legacy processes to be available for invocation, remove the endpoints of the process.

## Creating endpoints for processes

Endpoints that were created in the AEM Forms environment are no longer associated with imported processes that have a reset Deployment ID property. To invoke processes using the same type of endpoints, create them.

In AEM Forms, you create endpoints by adding start points to process diagrams. When the application is deployed, the start points cause the automatic creation of endpoints. You can add the following types of start points:

**Workspace:**

Users invoke processes from Workspace.

**Email:**

Users send an email to the AEM Forms Server to invoke the process.

**Mobile (Deprecated):**

Users invoke processes via AEM Forms express installed on mobile phones.

**Programmatic:**

Oracle™ Java™ and web service applications, REST, and applications built with Flex, can invoke processes.

Before you add start points, use Administration Console to review the end points that were used in the AEM Forms environment. Note the end point properties so that you can duplicate them in the start point properties. (See Managing Endpoints.)

**NOTE:** Workspace start points generate Task Manager end points. Before you can configure Workspace start points, configure assets and xml variables. (See Upgradinghuman-centric processes.)

Before you add a start point, you need to reset the Deployoment ID property. (See Breakingthe link to existing run-time instances.)

**To add a start point:**

1) Drag the start point icon onto the process diagram.
2) Select the type of start point to add and click OK.
3) Click the start point and provide values for properties in the Process Properties view.

## Replacing legacy subprocesses

If your process invokes legacy processes (subprocesses), import the legacy subprocesses into an application and reset the Deployment ID. The application that contains the subprocess to invoke must be deployed. When deployed, the service in AEM Forms for the process is created.

Before you replace a legacy subprocess, note the name and property values.

**Replace legacy subprocesses:**

1) Delete the icon for the subprocess from the process diagram.
2) Drag the Subprocess icon to the process diagram.
3) In the Find box of the Define Subprocess Activity dialog box, type the name of the process to invoke.
4) Select the process that exists in the application and click OK.
5) Select the subprocess icon and configure the properties in the Process Properties view.

## Configuring security settings

Security settings of legacy processes are not persisted when the Deployment ID property is reset. Configure security settings manually using administration console. (See Modifysecurity settings for a service.)

## Using deprecated operations

Some operations have been deprecated since the previous release. Typically, deprecated members are replaced with new ones. Replace deprecated operations in your legacy processes to take advantage of improvements they provide. Also, replacement operations are supported longer than deprecated ones.

**TIP:** In lists of services and operations, such as on the Services view, the names of deprecated services and operations have the suffix "(deprecated)".

## Exporting embedded documents from operation properties

In AEM Forms, some service operations allowed you to select documents from the file system to use as property values. After selecting the document, it became embedded in the process properties. For example, you could select DDX documents from the file system to configure the properties of invokeDDX and invokeOneDocument operations of the Assembler service.

AEM Forms now enables you to select documents from applications so that you can more easily update them. Consequently, after importing a process that includes embedded documents, you cannot use the Process Properties view to view or edit the documents. To view or edit embedded documents, export them to an application. After exporting, open the asset to view or edit it.

You can export embedded documents only after you import the process to an application. The process must also be checked out when you export. When you export the document, the operation property is automatically configured to use the new asset as a value.

**Export embedded documents:**

1) On the process diagram, select the operation that uses an embedded document as a property value.
2) In the Process Properties view, locate the property that uses the embedded document as a value.
3) Click the Export button .

*The Export button appears only when an embedded document value exists.*

4) In the File Name box, type a name for the file. Use the correct file name extension.
5) Select the application version and folder in which to save the file, and then click OK.

## Changing the reliance on null value results from XPath expressions

Review any logic in your processes that rely on the following situation:

- Null values are returned from XPath expressions.
- The XPath expressions retrieve values from data types that are defined by service components or custom Java classes in custom components (service container files).

In AEM Forms, this situation causes XPath expressions to return an empty-value instance of the data type instead of null. If your business logic relies on null values from XPath expressions, make sure the XPath expressions still return null. Otherwise, change the implementation of the business logic.

**NOTE:** XPath expressions that return values from xml variables continue to support null values.

## Upgrading human-centric processes

Human-centric processes that are imported into applications and have reset Deployment ID properties require several changes due to the following changes in LiveCycle ES2:

- Forms and form data are represented differently.
- Render and submit services are configured differently.
- Task Manager end points are created using Workspace start points.
- The User service is deprecated, and replaced with the User 2.0 service.

**NOTE:** Although the User service is supported, product changes have complicated its use in AEM Forms solutions. Adobe strongly recommends that you use the User 2.0 service.

Processes that use the User service or Task Manager end point require the following changes:

Replacingform-specific variables

ConfiguringWorkspace start points

Usingthe User 2.0 service

You must have already imported your processes (including custom render and submit processes and subprocesses), forms, and other resources into one or more applications. (See Importingresources, processes, and events to applications.) You also need to reset the Deployoment ID property. (See Breakingthe link to existing run-time instances.)

### Replacing form-specific variables

In AEM Forms, xfaForm, Document Form, and Form variables are not used to represent forms and form data. Instead, form assets are referenced directly and data is saved in xml variables. When forms submit PDF documents, the document is saved in document variables.

**NOTE:** Render and submit services are no longer configured in form variable properties. Instead, they are associated with the action profiles of form asset properties. Action profiles control which render and submit services are used, and other properties that involve presenting assets and data to users.

Before you replace an xfaForm, Document Form, or Form variable, note its property values. Perform the following procedure to create an xml variable for each xfaForm, Document Form, and Form variable in your process. You must have already imported the forms that the variables represented into applications.

*Create an xml variable:*

1) In the Variables view, click Create New Variable .
2) In the Name box, type a name for the variable, following these naming rules:
   – Must be a valid XML element name that contains only valid XML characters.
   – Must not start with a digit.
   – Must be less than 100 characters long.
   – Must be unique and therefore cannot be `id`, `create_time`, `update_time`, `creator_id`, or `status`, which are reserved variables always in the process data model.
3) In the Type list, select xml.
4) If the variable is used to store process data, select Process Variable in the Purpose area.
   – If the variable stores input data that is provided when the process is initiated, select Input.
   – If the variable stores data that is returned to the process initiator when the process is complete, select Output.
   – If the variable stores input data that is mandatory to initiate the process, select Required.
5) To import a schema to make the xml data structure appear in XPath Builder, click Import Asset.
6) Select an XSD file or an XDP file that has an embedded schema from an application and click OK.
7) Click OK.

### Using legacy render and submit processes

Action profiles provide more flexibility for prepopulating, rendering, and submitting presentation assets such as forms. The features of action profiles can negate the need for custom render and submit services. However, you can still use your custom render and submit processes from AEM Forms in your AEM Forms solutions.

**NOTE:** To determine whether your custom render and submit processes are still needed in AEM forms, see Designing data capture and presentation.

In AEM Forms, render and submit processes must include an input variable of type Task Context. To invoke your legacy render and submit processes directly, redesign them so that they use a TaskContext value as input. To avoid redesigning, create a process that invokes your legacy render or submit process:

- Legacy render and submit processes require no changes.
- Input values for the legacy process are configured on the service’s invoke operation.
- The TaskContext value that is automatically passed to the new process contains useful data for configuring the invoke operation.

Perform the following procedures for each form asset to use custom render and submit processes. You must have already imported the custom render and submit processes into an application. (See Importingresources, processes, and events to applications.)

*Reset Deployment ID properties:*

1) Reset the Deployment ID property of the legacy render and submit processes that you have imported. Resetting enables you to invoke them as subprocesses. (See Breakingthe link to existing run-time instances.)
2) Right-click the application that contains the render and submit process and click Deploy.

*Create new render and submit processes:*

1) In the Applications view, right-click the asset and click Manage Action Profiles.
2) Select the default action profile, or create an action profile to preserve the default one.
3) To create a render process, under Render Process click the Create A Render Process button .
4) Type a name for the process and click OK.
5) To create a submit process, under Submit Process click the Create A Submit Process button .
6) Type a name for the process and click OK.
7) Click OK to close the Manage Action Profiles dialog box.

*Configure render and submit processes:*

1) In the Applications view, right-click the new render or submit process and select Open.
2) Define the operation that was automatically added to the process:
   – For render processes, right-click the Render operation and click Define Operation.
   – For submit processes, right-click the Submit operation and click Define Operation.
3) Select the invoke operation of the legacy render or submit process and click OK.
4) On the Confirm dialog box, click Yes.
5) In the Process Properties view, configure the Input and Output properties of the invoke operation.

**Configuring Workspace start points**

Configure the Workspace start point that you added using the procedure in Creatingendpoints for processes.

Most of the values for the Workspace start point are the same that were used for the AEM Forms Task Manager endpoint. The following table shows how to use the values from the AEM Forms Task Manger endpoint properties to configure the Workspace start point.

| AEM forms Task Manager endpoint property | Corresponding Workspace start point property |
|---|---|
| Name | General/Name |
| Description | General/Description |

| AEM forms Task Manager endpoint property | Corresponding Workspace start point property |
|---|---|
| User Can Forward Task | Options/User Can Forward Task |
| Show Attachment Window | No equivalent property. If Permit Adding Attachments is selected, the attachment window appears. |
| Allow Attachment Adding | Attachment Options/Permit Adding Attachments |
| Task Initially Locked | Options/Task Initially Locked |
| Add ACLs for Shared Queues | Options/Add ACLs For Shared Queues |
| Task Instructions | General/Task Instructions |
| Process Owner | General/Contact Info |
| Categorization | General/Category |
| Operation Name | Not needed. The current process is invoked. |

Other Workspace start point properties configure the form and variable to use to store form data:

**Asset:**

The presentation asset to display to the user. Click the ellipsis button and select a form or Guide file. You can select a file from any application.

**Action Profile:**

The action profile to use with the asset. The action profiles that appear are already created for the asset that you selected.

**Variable:**

The variable in which to store the submitted data. Select either an xml, document, or Task Result variable:

- Select an xml variable if the asset submits field data that is en XML or XDP format.
- Select a document variable if the asset is a PDF form that submits a PDF document.
- Select a Task Result variable to save the field data as well as other information about the task.

**Reader Submit:**

Properties for enabling the submission of XDP or PDF forms that do not include a submit button. AEM Forms did not support this situation so you do not need to configure this property.

Before you configure the start point, configure your form assets and xml variables. (See Replacing-form-specific variables.)

*Configure the Workspace start point:*

1) In the process diagram, select the Workspace start point.
2) In the Process Properties view, configure the properties using equivalent values from the AEM Forms Task Manager endpoint according to the previous table.
3) Expand the Presentation & Data property group and provide the following values:
   – **Asset:** Select the form or Flex application to present to users.
   – **Action Profile:** Select the action profile of the asset that you want to use.
   – **Start Point Output:** Select the xml variable for saving the submitted data.

**Using the User 2.0 service**

Replace each Assign Task operation (User service) on your process diagram with an Assign Task operation from the User 2.0 service.

Most of the values for the new Assign Task operation are the same that were used for the deprecated Assign Task operation. The following table shows how to use the values from the deprecated operation to configure new Assign Task operations.

| Deprecated Assign Task property | Corresponding Assign Task operation (User 2.0 service) property |
|---|---|
| General properties | General properties |
| Route Evaluation properties | Route evaluation properties |
| Initial User Selection properties | Initial User Selection properties |
| Task Instructions | Task Instructions |
| Form Data Mappings | No corresponding properties. Forms and form data are represented differently in AEM Forms. |
| Task Access Control List (ACL) properties | Task Access Control List (ACL) properties |
| Delegation and Consultation properties | Reassignment Restrictions properties |
| Attachments and Notes/Show Attachment Window For This Task | Attachments/Show Attachment Window For This Task |
| Attachments and Notes/Do Not Initialize This Action With Any Notes Or Attachments | Attachments/Input List *(Do not specify an input list)* |
| Attachments and Notes/Copy All Notes and Attachments From Previous Task | Attachments/Input List *(Use the list variable that saved output attachments from the previous task)* |
| Attachments and Notes/Copy All Notes and Attachments From A List Of Documents | Attachments/Input List |

| Deprecated Assign Task property | Corresponding Assign Task operation (User 2.0 service) property |
|---|---|
| Attachments and Notes/Output Attachments | Attachments//Output Attachments |
| Routes And Priority/Initialize Task With Route Names | User Actions/Specify Custom Names For Actions. Add a user action for each route name. |
| Routes And Priority/User Must Select A Route To Complete The Task | No corresponding property. In AEM Forms, users must always select a user action to complete the task. |
| Routes And Priority/Select Priority For This Task | Priority/Task Priority |
| Reminders properties | Reminders properties |
| Escalation properties | Escalation properties |
| Deadline properties | Deadline properties |
| Custom Email Template properties | Custom Email Template properties |

Before you replace deprecated Assign Task operations, you must have already configured your form assets and xml variables. (See Replacingform-specific variables.) Use the following procedures to replace a deprecated Assign Task operation.

*Add an Assign Task operation:*

1) Drag the Assign Task operation to the process diagram.
2) In the Process Properties view, configure the properties using equivalent values from the deprecated Assign Task operation that you are replacing according to the previous table.
3) Expand the Presentation & Data property group
4) Select Use An Application Asset, click The ellipsis button next to the Asset box, and select the form asset.
5) Make sure the action profile that you want to use is selected.
6) (Optional) To prepopulate the asset, in the Variable list select an existing xml variable that contains the data. For example, select the variable that was used as the Start Point Output property value. To create an xml variable to use as the property value, click the Create New Variable button .
7) Expand the Output property group.
8) (Optional) To save submitted task data, in the Task Result list select a Task Result variable. To create a variable to use as the property value, click the New Variable button and create a Task Result variable.

   **TIP:** Task Result variables store all data that is submitted with tasks, such as asset data and information about the task.

9) (Optional) To save submitted asset data, in the Output Data list select an xml variable to save the data. Saving the asset data separately is useful for using the data to populate the asset of a subsequent task.

*Draw routes for the new Assign Task operation:*

1) If the routes that lead to or away from the deprecated Assign Task operation include conditions, note the expressions of the conditions.
2) Delete the routes that lead to and away from the deprecated Assign Task operation.
3) Draw routes to and from the new Assign Task operation that you added in the previous procedure.
4) (Optional) If the deprecated Assign Task operation used route names as task submission options, add a user action for each route. For example, if the route names were Approve and Deny, add user actions named Approve and Deny:
   – Expand the User Actions property group.
   – Click the Add A User Action button .
   – In the Action Name box, type the name of the action as you want it to appear to the user.
   – From the Destination list, select next operation to execute when the user action is clicked, and then click OK.
5) (Optional) If the routes to or from the deprecated Assign Task operation included conditions, add the conditions to the routes of the new Assign Task operation:
   – Select the route and in the Process Properties view, expand the Conditions property group.
   – Click the Add Route Condition button.
   – In the Expression box on the left, type the first part of the expression. If the condition is complex, click the ellipsis button to display the XPath Builder.
   – In the Operation list, select an operation.
   – In the Expression box on the right, type the second part of the expression. If the condition is complex, click the ellipsis button to display the XPath Builder.
   – Click OK to close the Route Properties dialog box.
   – If you have more than one routing condition in the Conditions category, select the join condition to determine how the conditions are evaluated:
     - Use AND Join For Conditions means that the route is valid only if all the conditions evaluate to `True`.
     - Use OR Join For Conditions means that the route is valid when one or more of the conditions evaluate to `True`.

     *The default join condition is Use OR Join For Conditions.*

*Delete the deprecated Assign Task operation:*

1) Right-click the deprecated Assign Task operation and click Delete Operation.

## Migrating LCAs to AEM Forms

***New for 10***

Workbench is integrated with the Archive Migration tool, which allows you to upgrade AEM Forms archive files to the application model. The tool was available as a plug-in for earlier versions, but is a standard feature of Workbench.

The Archive Migration Tool converts 8.x LCAs to the application model. The application model is a functional equivalent of the resources model seen with AEM Forms 8.x.

1) To access the Archive Migration Tool, go to File > Archive Migration Tool.
2) Click the Browse button and point to the LCA that you intend to migrate.
3) Enter a name and description for the new AEM Forms application. Click Finish.
4) In the applications view, you can now see that a new application has been created.

The newly created application has not been stored into the AEM Forms Server yet. You can accomplish the same by checking in the application and its assets in Workbench.

Perform the following tasks after migrating your LCA:

- When you migrate your LCA, references to process variables and activity configurations break. Sift through activities and ensure that activity configurations and process variables are intact. Update the references manually when ever necessary.
- Replace deprecated APIs with new APIs.
- If your processes use Events, redeploy the process and refresh the Events view.

**NOTE:** Archive Migration tool does not migrate custom components. Migrate them manually by reimporting them in the components view.

## Updating references to assets

After importing legacy processes, resources, and event types into applications, the URL of the item is changed. Update the references to the items in your forms (if the relative location has changed) and custom client applications.

### Process services

Change custom applications that you created with the AEM Forms SDK if they invoke services that are created for deployed processes. In AEM Forms, the name of these services in your Java™ code must be in the following format:

```
[Application Name]/[Service Name]
```

- *[Application Name]* is the name of the application that the process belongs to.
- *[Service Name]* is the name of the process service.

### Resources

Change references to resources to use URLs in the following format:

```
repository:///Applications/[Application Name]/[version]/[legacy path]
```

- *[Application Name]* is the name of the application that the resource belongs to.
- *[version]* is the version of the application.
- *[legacy path]* is the path of the resource as it appeared in the Resources view.

For example, in the Resources view, the path of a form is Loan/Forms/request.xdp. The form is imported into version 1.0 of an application named *NewCustomers*. The new URL is repository:///Applications/NewCustomers/1.0/Loan/Forms/request.xdp.

# 4. Using Workbench

This section provides an overview of information you need to consider before you begin working with Workbench.

## 4.1. Before you begin

### Process and form planning

Before you use Workbench, you should understand the requirements and the purpose of the processes and forms that you want to model, create, or automate. Be sure you understand the context in which your processes and forms will be used, how data will be captured, and how the data will flow through a process or between related processes or forms.

Depending on your organization, a business analyst or a project team may perform an analysis of existing processes and forms or may plan out the goals of the project. Often, an analysis or plan will map out the flow of a process or set of processes, identify the people involved in the process and determine what their roles are, and identify the kinds of forms to be used.

### Assembling a development team

Your organization also must determine who will be part of your development team. The team could include administrators, process developers, form developers, form authors, and possibly programmers to handle lower-level coding:

**Administrators:**

Install, monitor, maintain, and troubleshoot the environment. They import and export files using Applications and Services, and install, start, and stop services.

**Process developers:**

Work with business analysts to configure and map process requirements to Workbench functionality. They translate process definitions and implement processes in Workbench.

**Form developers:**

Develop and modify intelligent and interactive forms in Workbench, typically using scripting for tight back-end integration. These forms often support calculations and events.

**Form authors:**

Create and modify simple forms in Workbench. They may design parts of a form before handing it over to a form developer. They may also create fragments.

Each member on your team requires permissions and roles assigned to them. (See Settingaccess permissions.)

### Development process

When the planning and analysis is complete, your development team is assembled, and the computer environments are set up, you can start to use Workbench.

You can model, create, and automate processes and forms using Workbench. The development of a complete workflow solution can be a large and complex undertaking, and the specific steps your organization follows will be unique. However, many organizations follow these simplified generic steps, which are often simultaneous:

- Create the forms and, if necessary, define the data constructs that will pass information between steps in the process.
- Create a process diagram that links together participants, forms, and services, defining the overall life cycle of the process.
- Create any resources required by the forms or process diagram, such as common image files or data files.

  *When the solution is complete, it must be thoroughly tested. Do preliminary testing in the development environment, and then move the processes and forms to a separate testing environment for more thorough testing.*

## 4.2. Logging In to a AEM Forms Server

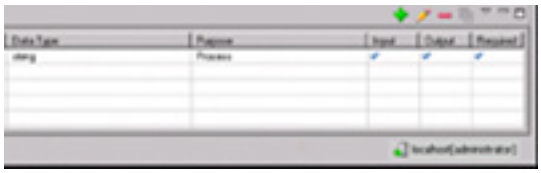
After you start Workbench, you establish a connection with a server by logging in. Your administrator must configure your user account and the server so that you can use Workbench.

### Logging in

To work on your processes or forms, you first must log in to a server. If a server has not already been configured for logging in, see Configuringserverconnections.

**NOTE:** If you have configured AEM Forms to use the Documentum or FileNet repository provider, and you want to log in to a repository other than that configured as the default in Administration Console, you should provide the user name as [username]@[Repository].

After you log in, the server that you are logged into and the user name that you logged in with appear in the lower right-hand corner of Workbench. The following illustrations shows that the administrator user is logged into the localhost server.



**To log in to a server:**

1) Start Workbench and select File > Login.

   **TIP:** You can also click the Login button .

2) In the login window, type your user name in the Username box and type your password in the Password box.
3) In the Log On To list, select the required server and then click Login.

**NOTE:** For security, the Reauthentication window may reappear at regular time intervals after you log in, at which time you need to provide your password again. Your administrator can determine the time interval. For more information, see "Configuring User Management > Configuring AEM Forms time-out setting" in User Management Help

.

## Increasing the memory allocation

If a low memory warning appears when you log in to Workbench, you can increase the amount of memory allocated to Workbench.

To configure the allocated memory, modify the workbench.ini file property that controls the amount of allocated memory. The property is in the format `-Xmx`*[number]*`M`, where *[number]* is the number of MB of memory that is allocated. The default amount of memory that is allocated is 512, which is set by using the property value of `-Xmx512M`. You increase the amount of memory in increments of 256. For example, if the warning appears when 512 MB of memory is allocated, increase the allocated memory to 768. If the warning occurs again, increase to 1024.

The workbench.ini file is located in the *[install directory]*/Adobe/AEM forms Workbench/Workbench directory, where *[install directory]* is the location where you installed Workbench.

**Increase the allocated memory:**

1) In a text editor, open the workbench.ini file.
2) Locate the line that contains the property that sets the amount of allocated memory, and modify the values of the property as required. For example, `-Xmx768M` allocates 768 MB of memory.
3) Save and close the workbench.ini file and restart Workbench.

If a low memory warning appears after extended use of Workbench, close and restart it.

## Configuring server connections

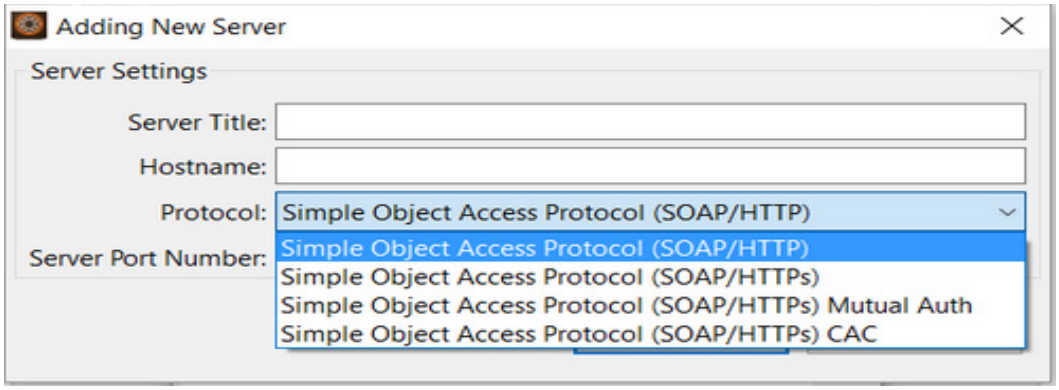
You need to configure a connection to the AEM Forms Server that you want to log into and access the resources that it stores. If a server is not already configured or you want to change the server settings, you can configure the server from the login dialog box. Ask your administrator for the values you should use when configuring a server.

After you configure a server, you can select the configuration in the login window.

**To configure a server:**

1) In the AEM Forms login dialog box, click Configure. The Manage Configured Servers dialog box appears.
2) To add a new server, click the Add New Server button . To edit an existing server, select a server and click the Edit Selected Server button .
3) In the Server Title box, type a name to describe the server. This name is descriptive only. It is displayed in the selection menu of the login window.
4) In the Hostname box, type the name or IP address of the AEM forms Server.
5) In the Protocol list, select Simple Object Access Protocol (SOAP/HTTP), Simple Object Access Protocol (SOAP/HTTPs)CAC, Simple Object Access Protocol (SOAP/HTTPS), or Simple Object Access Protocol(SOAP/HTTPs) Mutual Auth.

   Add forms server URL and name.




6) In the Server Port Number box, type the port number provided by your administrator. This specifies the port number used to connect to the server.
7) Click OK, and then click OK again.
8) On the server, In the login window, verify that you selected the correct server from the Log On To list.

   Workbench Login screen.

*If you use the **Simple Object Access Protocol (SOAP/HTTPs) Mutual Auth** option to connect to the server, only then the following options appear on the login screen:*

a) Key Store: Path of the key store file. The file resides on the machine which has Workbench installed. Specify the path in `file:[path_of_the_file]` format.

b) Key Store Password: Password of the key store. In general, you create key store passwords while creating the key store.

c) Trust Store: Path of the trust store file. The file resides on the machine which has Workbench installed. Specify the path in `file:[path_of_the_file]` format

d) Trust Store Password: Password of the trust store. In general, you create trust store passwords while creating the trust store.

If you use the **Simple Object Access Protocol (SOAP/HTTPs) CAC** option to connect to the server,before logging in to AEM Forms server, insert the CAC card to the machine running the Workbench. Also, ensure that AEM Forms server are enabled for CAC and Mutual Authentication.

When you press login, after inserting CAC card to the machine, a dialog with all the available certif-icates is displayed. Select a certificate and press OK. The workbench is connected to the AEM Forms server. The connection remains active as long as the CAC card is in place.

## Logging out

If you are finished using Workbench or want to log in to a different server, you need to log out of the server you are connected to.

**To log out of a server:**

1) Select File > Logout.
2) In the logout window, click Logout.

   **TIP:** You can also click the Logout button .

## User permissions

To log into Workbench, your user account needs to be assigned the role of an Application Administrator. This role provides the following permissions that are relevant to Workbench users.

- Use administration console (Applications and Services pages) to configure service run-time properties; add, remove, and configure endpoints, and configure security.
- Install and uninstall components.
- Create applications.
- Delete applications
- Create event types.
- Delete event types.
- Create processes.
- Delete processes.

- See a list of recorded process instances and play them.
- Start and stop services, and activate and deactivate processes.
- Open processes for reading and editing.
- Invoke services.
- Use Workbench features that enable you to create forms:
- Set and delegate access permissions for applications.

For more information about assigning roles to users, see AEM Forms administrationhelp

or contact your AEM Forms administrator.

## Mutual authentication

You can configure mutual authentication between Workbench and AEM Forms server. The mutual authentication requires key store and trust store pair and public key and private key pair to reside on Workbench and AEM Forms server. The public certificate of the workbench is imported into AEM Forms server and the public certificate of the AEM Forms server is imported into Workbench. The key store contains the private keys. Do not share these keys with anyone.

The following list of commands helps you create a simple self-signed certificate and configure it under workbench/server:

1) Create public/private key pair for AEM Forms server:

   ```
   keytool -genkey -alias teiid -keyalg RSA -validity 365 -keystore
   server.keystore -storetype JKS
   ```

2) Export the public certificate server.cert. This server certificate is imported to Workbench:

   ```
   keytool -export -alias teiid -keysto. re server.keystore -rfc -file
   server.cert
   ```

3) Create public/private key pair for Workbench:

   ```
   keytool -genkey -alias cteiid -keyalg RSA -validity 365 -keystore
   clientnew.jks -storetype JKS
   ```

4) Export the public certificate client.cert file. This client (Workbench) certificate is imported to AEM Forms:

   ```
   keytool -export -alias cteiid -keystore cliennew.jks -rfc -file
   client.cert
   ```

5) Import public certificate of server into trust store of workbench:

   ```
   keytool -import -alias teiid -file server.cert -storetype JKS -keystore
   client.truststore
   ```

6) Import public certificate of workbench into trust store of server:

   ```
   keytool -import -alias cteiid -file client.cert -storetype JKS
   -keystore server.truststore
   ```

# 4.3. Working with Applications

In AEM Forms, an *application* is a container for storing assets that are required for implementing a AEM Forms solution. Examples of assets are form designs, form fragments, images, processes, DDX files, form guides, HTML pages, and SWF files. You can use applications for two purposes:

- As a container for the related assets that are required for managing an application life cycle. For example, you have a Loan Approval application that contains a form design, a company logo, fragments the form design uses, and the loan-approval process.
- As a container for unrelated assets that are used in different applications and by various developers. For example, you have a project that contains the various form designs your organization uses and the images these form designs use.

When you first create an application or an asset, it is stored in a folder on your local computer. To make it available to other application developers, check it into the AEM Forms repository. The applications and assets that are stored in the AEM Forms repository are available to developers on an as-needed basis. This means that a process developer can easily access any components that are required to develop a AEM Forms application.

When the application is completed, you can deploy it directly from the Applications view so that it can be invoked. The deployment process can be repeated as required to update the application in the repository.

## About the Applications view

The Applications view in Workbench shows applications that you created or retrieved from the server. For each application, the view shows all files (assets) in a single, multilevel tree. The root nodes in the view represent applications. The first level child nodes represent application versions.

Icons for the applications and assets can have the following markers attached to them to indicate a particular state:

- Marks a new asset that exists only in your local folder. This asset is not checked into the repository.
- Marks an asset that is checked out of the repository.
- Marks a deployed application.
- Marks an asset for deletion. The asset will be deleted when you perform a check in operation.
- Marks an asset that has validation warnings. Open the Validation Report view to see the list validation messages.
- Marks an application or an asset that reports a problem. The appearance of this mark is turned off by default. To turn it on, select Window > Preferences > General > Label Decorations, and then select Validation Decor option
- Marks an asset imported from LiveCycle ES (8.x).
- Marks a process that has recording enabled.

By default, the Applications view displays the applications and assets that are on your local computer. You can also view the applications and assets that either you or other users checked into the repository.

To view these applications and assets, click the Show Remote Resource icon. To return to the view of your local folder, click the Show Local Resources icon.

When showing local resources, you can perform the following tasks in the Applications view:

- Add and remove applications
- Add and remove assets
- Organize assets
- Edit and view assets
- Deploy applications

When showing remote resources, you can perform the following task in the Applications view:

Managinguser access to applications

## Developing applications in a multi-developer environment

When working in a multi-developer environment, it is recommended that you follow these procedures to ensure that you do not override changes that other users made:

- Each developer must log in to Workbench with a unique user ID.
- Use the Check In/Check Out system to ensure that only one person at a time works on an asset and that the latest version of an asset is available to other users.
- Synchronize your applications and assets before you check them out. Synchronizing copies the last check-in version of the application or asset from the server to your local computer. This procedure ensures that you are always editing the latest version of the application or asset.
- Because Workbench does not support merging of multiple changes, make sure that only one user at a time edits the assets.

### Synchronize the local version with the version in the repository

1) Right-click the application or the asset and select Synchronize.

### Checking in applications or assets

When you first create an application or an asset, it is created on your local computer. To make it available to other developers for editing or including in other applications, check the application or the asset into the server repository. You can check in individual or multiple applications or assets at the same time.

A warning dialog box is displayed when you attempt to check in an asset with modified process configuration parameters. Checking in an asset with modified configuration parameters can affect running process instances. Click Yes to continue with the check-in, or click No to cancel the check-in.

Also, you must check in the application to deploy it. (See Deployingapplications).

*Check in assets*

1) Use one of these methods to check in the assets:
   – Check in the application that contains the assets.

- To check in a single asset, right-click the asset and select Check In. When prompted, click Yes to proceed with the check-in operation.
- To check in the assets that are in a folder, right-click the folder and select Check In.
  - In the Check In Assets dialog box, you can select all assets to be checked in, some assets to be checked in, or none of the assets to be checked in. Click OK.
  - When prompted, click Yes to proceed with the check-in of the changed assets.
- To check in multiple assets that are located in various parts of the application and/or in different applications, select the assets by using Ctrl+click or Shift+click, and then right-click one of the assets and select Check In. When prompted, click Yes to proceed with the check-in operation.

*A status dialog box appears after the check-in operation is complete.*

*Check in applications*

1) Right-click an application version and select Check In. To check in multiple applications, use Ctrl+click to select the applications, and then right-click one of the selected applications and select Check In.
2) (Optional) If the application contains the assets that are not checked in, select one of the following options:
   - **Check In All Files:** Workbench checks in all the assets.
   - **Check In Selected Files:** Workbench checks in only the assets that you explicitly select.
   - **Check In None Of The Files:** Workbench does not check in any assets.
3) (Optional) In the list of the checked-out assets, select the assets to check in. Use Ctrl+click or Shift+click to select multiple assets.

**Checking out applications or assets**

To edit an application or an asset, check it out of the repository. A checked-out application or asset is available for editing by the user who checked it out.

If you want to check out an application or asset that another user has checked out, you can forcibly check it out. This action discards any changes that the other user made after their checkout.

*Check out applications or assets*

1) Right-click an application or an asset and select Check Out. To check out multiple applications or assets, use Ctrl+click to select them, and then right-click one of the applications or assets and select Check Out.
2) If another user has any of the assets checked out, select one of the following options:
   - **Force Check Out On All Files:** Workbench forces the checkout of all assets.
   - **Force Check Out On Selected Files:** Workbench forces the checkout only for the assets that you explicitly select.
   - **Force Check Out On None Of The Files:** Workbench does not force the checkout of any assets.

3) (Optional) In the list of assets that another user checked out, select the assets to forcibly check out. Use Ctrl+click or Shift+click to select multiple assets.

### Discarding the changes

After you check out an application or asset and edit it, if you need to discard the changes, use the Revert command to quickly return to its checked-out state. Using the Revert command for the application discards changes made to all its assets since the last checkout.

**NOTE:** The Revert command cannot be used to revert to a previous version of the asset.

*Discard changes made to an asset:*

1) Right-click the asset and select Revert.

### Viewing asset history

To view the history of asset revisions, use the History command; it displays the list of checked-in versions of the asset.

*Review the history of an asset version:*

1) Right-click the asset and select History.

### Managing user access to applications

You must have the Application Administrator role assigned to your AEM Forms user account in order to develop applications. In this role, you can assign limited access to the application development environment for users who have other roles assigned. You can permit other users to have the following type of access:

**Read:**

The user can view the application.

**Delegate:**

Although the user is not the Application Administrator, that user can extend read/write permissions to another user.

**Write:**

The user can modify the application and save the changes.

For example, if a contractor is assigned to your project, you can grant that person Read and Write permissions only for the applications that the person will be working on.

*Assign access permissions for an application:*

1) In the Applications view, click Show Remote Resource to view the list of applications on the server.
2) Right-click the name of the application for which to configure user access and select Manage Access.

3) Click Add and in the Select User(s) And Group(s) dialog box, specify how to search for the users:
   – To search by name, select User Name.
   – To search by the user's or group's email address, select Email.
4) In the box, type all or part of the user name or email address and click Find. Type no characters in the box and click Find to retrieve a list of all users.
5) In the Results pane, select a user or a group and click Add.
6) Click OK to return to the Access Control dialog box.
7) In the Users and Groups area, select a user or a group and, in the Permissions area, select the permissions to assign to that user or group.
8) Repeat step 7 for every user or group for which to configure access to the application.

RELATED LINKS:

Workingwith Applications

Addingand removing applications

Workingwith assets

Deployingapplications

Userpermissions

## Adding and removing applications

Whether implementing a AEM Forms solution or creating a container for the assets that are shared among AEM Forms developers, first create an application in the Applications view in Workbench. Use the New Application wizard to create an application.

You can also copy an application from the server and use it as a start point. This method creates a local copy of the server application. If you edit and check in the application, these changes are saved in the original application.

When you no longer need an application, you can remove it. You have a choice of removing only the local copy or both the local copy and the copy stored in the repository.

### Creating an application

The New Application wizard guides you through the steps of adding a new application to the Application view. You can later modify most of the information entered in the wizard.

The wizard creates the application in your local folder. To share development of the application with other developers and before you deploy the application, check it into the repository.

When creating an application name, follow these rules:

- Alphanumeric characters, double-byte characters, and spaces can be used.
- These characters cannot be used: /\:+$?%*: |"<>.[]TAB.
- Control characters (ASCII value less than 32) cannot be used.
- Maximum number of characters and spaces is 40.

**NOTE:** Full path to an asset cannot exceed 256 characters. This path includes the entire path where the asset is cached on the computer.

*Create an application using the wizard:*

1) Select File > New > Application.
2) In the Application Name box, type a unique name for your application.
3) (Optional) In the Description box, type a brief description of the application. You can add or change the description later by right-clicking the application version and selecting Properties.
4) Click Finish.

### Adding applications from the server

To work on the application that another developer created, first add the copy of that application to your local folder. Then, check out the application or its assets to modify them and check in the modified components.

To determine which applications you already have locally on your computer use the Show Local Applications checkbox in the Get Applications dialog box to include or exclude applications from the list.

*Add an application from the server:*

1) Select File > Get Application.
2) (Optional) To include local applications in the list of available applications, select Show Local Applications.
3) From the list of applications in the repository, select the application to add to your local folder and click OK.

### Removing applications

When you delete an application, all the assets associated with it are also deleted. By default, an application is deleted from the local folders. If necessary, you can specify to also remove the application version from the server.

*Remove an application:*

1) In the Applications view, right-click the application version and select Delete.
2) To delete both the local and the server version of the application, select Also Delete Application From The Server.
3) Click Yes to confirm the deletion.

RELATED LINKS:

Workingwith assets

Applicationsand assets versioning

## Working with assets

Components that make up the application are called *assets*. Any file that is required to run the application when you deploy it must be added to an application. You can also include assets that are not required for running the application but may be added later or used in another application. For example, you can create an application that acts as a library of assets for other applications. Examples of assets are processes, form designs, images, and DDX files.

Your application can also contain assets that you created in AEM Forms. Their original properties are preserved provided you use the asset within the application that it was imported into. However, when you copy such an asset into another application, it becomes a AEM Forms application asset. (See Leveraging legacysolutions in AEM Forms.)

**IMPORTANT:** To avoid problems with application deployment and execution, manually remove the Deployment ID from LiveCycle ES (8.x) assets before you reference them in AEM Forms. Alternatively, migrate the LiveCycle ES (8.x) assets using the Archive Migration Tool before you use them in AEM Forms.

When working with application assets, you can perform the following actions:

- Add and remove assets
- Organize assets
- Edit and view assets

### Adding and removing assets

Use one of the following methods to add assets to your application:

- Create an asset in Workbench and specify the application that the asset belongs to.
- Copy assets from another application.
- Add assets created outside Workbench, such as graphic files or text files.

When you first create an asset, it is created in your local folder. To make it available on the server, check in either the application that the asset resides in or the particular asset.

*Creating assets in Workbench*

The New Asset wizards that you follow when creating the following assets require you to select the application the asset belongs to:

- Process (See Creatingprocesses.)
- Form (See CreatingForms.)
- Assembly Descriptor (DDX document) (See GettingStarted with (Deprecated) Document Builder.)
- User List (See Creatingand changing user lists.)
- Component Reference (See Adda Component Reference asset to the application)
- Event Type (See Createcustom event types.
- Guide (See CreatingGuides.)

Create an asset:

1) Use one of the following methods to select the type of asset to create:
   – Select File > New > *asset type*, where *asset type* is the name of a valid asset type.
   – Select File > New > Other and select the asset type from the list.

2) Follow the instructions in the wizard.

**NOTE:** When you select File > New > Other, the Enter Or Select The Parent Folder box displays the tree structure of the local file system. It does not display the Workbench Applications view.

When creating an asset name, use the following rules:

- Alphanumeric characters, double-byte characters, and spaces can be used.
- These characters cannot be used: /\:+$?%*: |"<>.[]TAB.
- Control characters (ASCII value less than 32) cannot be used.
- Maximum number of characters and spaces is 40.

**NOTE:** Full path to an asset cannot exceed 256 characters. This path includes the entire path where the asset is cached on the computer.

*Copying assets from another application*

The application you copy assets from must be available on your local system. Therefore, if the source application is on the server, first add the application to your local view by using the Get Application command.

Copy assets between applications:

1) (Optional) Select File > Get Application and select the application to add to your local system.
2) Expand the application tree in the Applications view and select the asset to copy.
3) Use one of the following methods to copy the asset:
   – Select Edit > Copy.
   – Click the Copy button.
   – Right-click the asset and select Copy.
   – Press Ctrl+C.

4) In the destination location, select either a folder or the application version and use one of the following methods to paste the asset:
   – Select Edit > Paste.
   – Click the Paste button.
   – Right-click the destination folder or application version and select Paste.
   – Press Ctrl+V.

   **TIP:** You can also drag the asset from one location to another while pressing the Ctrl key.

**NOTE:** Assets can reference other assets. When an asset is copied to a different location (such as another folder or application), any references still point to the old location. If for any reason the reference to the old location is broken, the process at the new location fails at runtime. (For example, if the asset at the

old location is deleted.) To avoid this issue, after you copy an asset to a new location, update any references.

*Adding assets created outside AEM Forms*

Your application can contain assets that were created in another application. For example, if your process displays illustrations on the forms or uses a spreadsheet, these files must be included in the application.

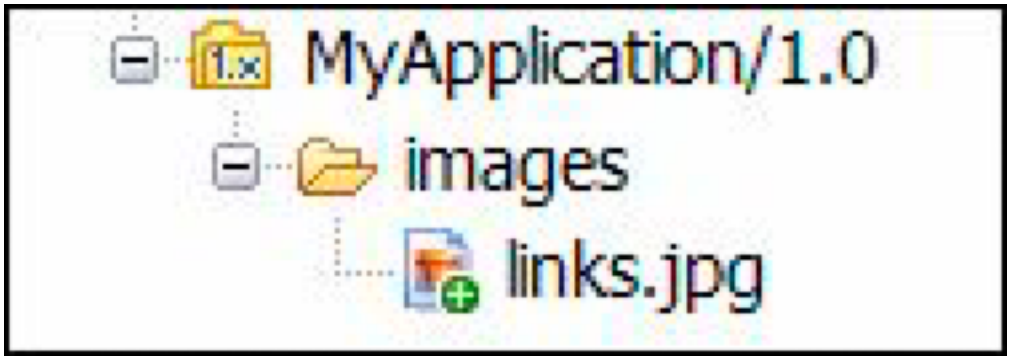
Add assets created outside AEM Forms:

1) Drag the file from the desktop or the folder to the application or the application folder.

*Removing assets*

You can easily remove assets that are no longer required in the application. When deleting assets, remember these situations:

- Assets that exist only in your local folder, and are not checked into the repository, are deleted from the application immediately. The local asset icons in the application tree have a white cross on a green background in a lower-right corner.




- Assets that were checked into the repository at least once are marked for deletion and are deleted from the application after you perform a check in operation. The icons for the assets marked for deletion are annotated with a red X.

Delete an asset:

1) Right-click the asset and select Delete.
2) Click Yes to confirm deletion. A local asset is immediately removed from the list. An asset that was checked into the repository is marked with a red X.
3) Right-click the asset that is marked with a red X and select Check In. The asset is removed from the local list and from the repository.

### Organizing assets

Assets can reside below the root of the application in a single list. Alternatively, you can organize them into logical groups inside folders and subfolders. Create a folder structure inside your application to suit your particular requirements. For example, you can group all the forms used in the process in a folder called *Forms*.

As you can create only one folder level at a time, repeat the procedure several times to create a deeper folder structure.

***TIP:*** *You can add various assets to a AEM Forms that includes image files, DDX files, processes, and forms. It is recommended that each application contains 50 assets or less. For existing applications with 50 or more assets, partition them into smaller applications. When you keep the number of assets small, it improves application performance and maintainability.*

*Create a folder*

1) Select File > New > Folder.
2) On the New Folder panel, select the application version and specify the name of the folder.

**NOTE:** You can also create a folder while adding assets to the application.

When creating a folder name, use the following rules:

- Alphanumeric characters, double-byte characters, and spaces can be used.
- These characters cannot be used: /\:+$?%*: |"<>.[]TAB.
- Control characters (ASCII value less than 32) cannot be used.
- Maximum number of characters and spaces is 40.

**NOTE:** Full path to an asset cannot exceed 256 characters. This path includes the entire path where the asset is cached on the computer.

*Add an asset to a folder*

1) Use one of the following methods:
   – Select the folder name while creating an asset.
   – Drag an asset from the desktop to the folder.
   – Copy and paste assets between folders.

*Copy and paste an asset between folders*

1) Select an asset and use one of the following methods to copy it:
   - Select Edit > Copy.
   - Click the Copy button.
   - Right-click an asset and select Copy.
   - Press Ctrl+C.
2) Select the destination folder and use one of the following methods to paste the asset:
   - Select Edit > Paste.
   - Click the Paste button.
   - Right-click the destination folder and select Paste.
   - Press Ctrl+V.

   **TIP:** You can also drag the asset from one location to another while pressing the Ctrl key.

*Renaming assets*

You can rename assets for undeployed applications. When you rename the asset, you must change all references to the asset. For example, if you rename a XSD file in your application, you must change the form that references the XSD file to use the new name.

1) Right-click the application where the asset exists and select Undeploy. You need to complete this step only when your application is deployed.
2) Right-click the asset to rename and select Check Out.
3) In the Rename dialog box, type new name for the asset and click OK.
4) Right-click the asset and select Check-In.
5) (Optional) Check-out processes and forms that use the asset that you rename and redeploy the application.

### Editing and viewing assets

After an asset becomes part of an application, you can view it, edit it, and change its properties. Some assets open inside Workbench, and other assets you open in their native programs. When opening an asset, you can either let Workbench select a default editor, or you can manually select the editor.

If you are opening an asset to edit it, you must first check it out. When you start editing an asset that is not checked out, an automatic checkout message prompts you to check out the asset before you save it. If you decline the checkout, the asset remains in read-only state. Use a different name to save it.

You close an asset when you no longer want to view or edit it. An asset that you edit remains checked out after you close it. You must check it in after you edit it to allow others to modify it.

You can also edit several properties for an asset, such as description, custom metadata, and deployment information.

*Open an asset for viewing or editing:*

1) Use one of the following methods:
   – Right-click the asset and select Open or Open With > *editor name*. This command opens the asset in its default editor.
   – Right-click the asset and select Open With > Other. This command opens the Editor Selection dialog box, where you can select the editor to use for editing the asset. You can select the editor from the list of internal editors or the list of external programs. If you cannot find the required editor on either list, you can browse for the location of the specific editor.

2) To always automatically check out an asset, in the Automatic Checkout dialog box, select Never Show This Dialog Again And Remember My Decision. To return to the default setting and see the Automatic Checkout dialog box each time you checkout an asset, click Window > Preferences > AEM Forms > Preferences, and deselect Perform Asset(s) Checkout Automatically.

*Edit asset properties:*

If the asset has custom properties defined, they are listed in the Custom Properties table.

The Deployment ID and Deployment Version boxes display the name, path, and version of the assets that were imported from the LiveCycle ES (8.x) application. (See Leveraginglegacy solutions in AEM Forms.)

1) Right-click the asset and select Properties.
2) (Optional) In the Description box, modify or add a descriptive text for the asset.
3) (Optional) To edit a custom property, select the property in the Custom Properties table, click Edit , and change the value.
4) To remove the values for Deployment ID and Deployment Version, click Reset.

RELATED LINKS:

Addingand removing applications

Developingapplications in a multi-developer environment

### Managing asset dependencies

Dependency is a functional relationship that an asset shares with another asset. An asset can have outgoing references by referencing another asset or an asset can have incoming references when dependent assets reference it. Modifying assets and asset properties may have adverse effects. The dependency detection feature in Workbench notifies when such modifications are attempted. The following are instances when dependencies are detected:

- **Creating archive files**: When creating an LCA, there may be outgoing references to assets outside the application. Such assets are implicitly selected on the Archive Properties dialog box to be archived along with the selected application. You can choose to omit these outgoing references by deselecting appropriate assets. For more information, see Creatingarchive files.
- **Adding applications from the server**: You can choose to add assets that have outgoing references from the application. For more information, see Adding applicationsfrom the server.
- **Deleting an asset**: Workbench throws a warning when you try deleting an asset that has incoming references. Ensure that you do not delete such assets that are critical for dependent assets to be functional.

- **Renaming an asset**: Workbench throws a warning when you try renaming an asset that has incoming references. If you intend to persist with these dependencies, ensure that you have the Update References checkbox checked on the Renaming dialog box.

*Visualizing dependencies*

Dependency Visualisation feature allows you to track and monitor all incoming and outgoing references of an asset. Use this feature as an indicator of the dependencies of the asset before you modify its properties.

1) In Workbench, access **Window** > **Show View** > **Other** to open the Show View dialog box. Drill down General views to open the Properties view.
2) Select an asset in the Applications view whose references you want to view.
3) In the Properties view, click Outgoing References to view the list of dependencies of the asset. Also, click Incoming References to view the list of assets that are dependent on the selected asset.

**NOTE:** You can view only the immediate level of asset references. For example, consider that ProcessA has an outgoing reference to ProcessB and Process B in turn has an outgoing to ProcessC. When viewing outgoing references of ProcessA, ProcessC is not listed as a reference.

## Applications and assets versioning

Versioning is a quick way of creating an application that only partially differs from another application. A new application version is an exact copy of the original application, distinguished only by its new version number. When the new version is created, you can modify its structure and assets. You can create, maintain, and deploy multiple versions of the same application.

The *application version* is a decimal number in which the first digit represents the major version and the second digit represents the minor version (for example, 2.1). This unique decimal number distinguishes the application version from other versions. A *major version* number is usually changed when you make substantial changes to the application. Smaller changes usually require the change in the *minor version*. A new application is automatically assigned version 1.0. The new version of the application is automatically checked into the repository. You check it out to modify it.

**IMPORTANT:** The Run As security setting that you configure for an application in the administration console is not retained when you create a version of the application. You must reconfigure this setting for the new version of the application.

### Asset versions

Assets are not versioned separately but derive their version from the hosting application's version. When you create a version of the application, it contains copies of all the assets from the original version.

**NOTE:** When creating an application version that will host a process developed in LiveCycle ES (8.x), specify the application version to be higher than the process version. For example, if the process has three versions, 1.0, 1.1, and 1.3, the default version of the application should be 1.4.

At run-time, the process version determines the run-time service version. Incoming service requests use the highest version of the service.

If your application contains a process that is used as a sub-process in another application, the highest deployed version of this process will be used. For example, application A has two versions deployed, 1.0 and 2.0. Application B uses the process from application A (called *procA*) as a sub-process. At run time, application B runs procA from the 2.0 version of application A. Furthermore, if version 3.0 of application A is deployed, new instances of application B runs this version of the process. The currently running instances of application B continues running version 2.0 of the process.

**NOTE:** When invoking services programmatically, you can optionally invoke a specific version of the service. Otherwise, the highest deployed version is invoked.

When you create a new version of a process, update the following elements manually:

- After you have updated a new version of an application, ensure that all processes and their sub-processes make reference to the correct version. In some cases, subprocesses reference the original version of the application and not the new version. To update a reference, delete the icon for the subprocess from the process diagram and add the correct subprocess. For more information, see Invokingsubprocesses.
- An archiving operation that references a document in a local application, if the operation was added using Archive Wizard. Creating a new version of a process breaks this reference. Manually update the reference in the operation properties.

*Create a version of the application:*

1) In the Applications view, right-click the top-level application folder and select New > Application Version.
2) In the Select The Application list, select the application version to create a version from.
3) (Optional) Specify the Major and/or Minor Version numbers.
4) (Optional) In the Description box, type the text that describes this version of the application.

**NOTE:** Update the path of a reference to a .swf file in the referencing application manually, when you increment the source application version as the references are not updated automatically. For example, myApplicationA/1.0 and myApplicationB/1.0 are two applications. myApplicationB/1.0 references a swf file, myApplicationA/1.0/swfTest.swf. Assume myApplicationA/1.0 is incremented to myApplication/1.1. It is observed in myApplicationB/1.0 that the path of the swf file is not updated and remains myApplicationA/1.0/swfTest.swf.

## Deploying applications

Before you invoke an application, make it available in the runtime registry by deploying it. The application and the assets you want to include in the deployment must be checked in. You can deploy one or more applications at a time.

After the initial deployment, if you change parts of the application, you do not need to redeploy the entire application. Redeploy only the parts that are changed. When redeploying an updated long-lived process, the currently running process follows the updated steps in the new process if they are not already completed.

You can also undeploy the application. This procedure removes the application from the runtime registry, therefore making it unavailable for invoking. Use this approach to stop the application from

running and to keep it in the repository. Undeploying is different from deleting because deleting also removes the application from the repository.

**NOTE:** Make sure that no instances of the process are running when you undeploy or delete the application. Otherwise, these instances will not complete.

**NOTE:** Undeploying the application deletes configurations that were done for the application in administration console. Therefore, when you deploy this application again, you must reconfigure it.

**TIP:** To check whether the application is deployed, pause the pointer on the name of the application in the Applications view. A tooltip shows either Not Deployed or Deployed. Also, the deployed applications use this icon in the applications tree.

### Deploy an application

1) Right-click the application version and select Deploy.
2) If assets that are not deployed exist in your application, in the Check In Assets dialog box, specify how to proceed by selecting one of the following options:

**Check In All Files:**

Workbench checks in all the files.

**Check In Selected Files:**

Workbench checks in only the files that are selected for check-in.

**Check In None Of The Files:**

Workbench does not check in any files.

**NOTE:** To select multiple files for check-in, press either Ctrl or Shift and click the files in the list.

### Redeploy an application:

1) Right-click the application version and select Deploy.
2) Select Checkin All Files or Checkin Selected Files and specify the files that require check-in and deployment.

### Undeploy an application:

1) Right-click the application version and select Undeploy, and then click OK in the Warning message.

RELATED LINKS:

Workingwith Applications

Addingand removing applications

Addingand removing assets

Developingapplications in a multi-developer environment

## Moving applications into another environment

At some point in your application development cycle, you may need to move the application to another location for one of these reasons:

- You want to test your application in a dedicated testing environment.
- You want to temporarily move your application to another development environment, such as a laptop.
- Another developer is taking over your application development.
- You want to deploy your application in a live production environment.
- You need to create a backup of your application.

To move your application from one environment to another, use a AEM Forms archive file. A AEM Form-s*archive file* is a compressed file that contains one or more applications and assets.

Moving applications and assets to another environment involves these tasks, which are performed in the following order:

**Task 1. Create archive file:**

In Workbench, create or update an archive file to include applications and assets you want to move. (See Creatingarchive files.)

**Task 2. Import archive file:**

In administration console, use the archive file to import applications and assets into another location. (See “Managing AEM Forms Archives” in AEM Forms administrationhelp

.)

These tasks may be repeated if an archive file is moved from system to system or if an application is updated repeatedly and the updates need to be reflected in the other environment.

### About archive files

An *archive file* is a compressed file that contains selected applications and assets you want to move.

An archive can also contain assets that are not related to any processes.

You can include as many applications and assets as you need in the archive file.

In addition, the *archive file* contains the data (*as CQ packages*) to be exported to the AEM Forms CRX repository.

The filename extension of an archive is .lca.

**NOTE:** External resources are automatically included in an archive file only if they are referenced by the Form variables. Resources referenced by the operation properties are not automatically included but can be added to the archive manually.

### Creating archive files

When creating an archive, you have these options:

- You can start with an empty file and select all the applications and assets to include in the archive.
- You can start with a previously created archive file and modify its content by adding or removing applications and assets. This method is a time-saving method to use when the new archive differs from the existing one only by a few items.
- You can create a patch for the already imported archive. This method is useful when you created an archive, imported its content to another location, but now you have some assets that are updated and need to be reimported. You can create an archive that contains only the assets that need to be updated.

Before you create an archive file, check in the applications and assets to include in the archive. If your application contains assets that are checked out, Workbench prompts you to check them in before you start archive creation.

The following user permissions are required for creating an archive file.

- To include processes in the archive, the user must have Service Read and Service Invoke permissions.
- To include resources in the archive, the user must have Repository Read and Repository Traverse permissions for the resource being exported, and Repository Traverse permission for the parent folders.

If you are archiving an application that uses a custom component, add the Component Reference asset to the application. This asset provides a way to explicitly declare a relationship for a component.

*Add a Component Reference asset to the application*

A component reference provides a way for you to explicity declare a relationship to custom component that is used by your application. For more information about custom components, see "Developing Components
".

1) Select File > New > Other.
2) In the Select a Wizard panel, select Component Reference and click Next.
3) In the Name box, specify the name for the component reference asset.
4) (Optional) In the Description box, type a brief description of the asset.
5) In the Location area, specify the path to the asset location within the application hierarchy. The path must include the name and version of the application and, optionally, subfolder names. An example of a valid path is /MyApp/1.0/processes. You can either type the path or select the required location from the application tree.
6) (Optional) Click New Folder to create a subfolder in the application tree.
7) Click Next, click Browse, select the component, and then select Finish to create the Component References asset.

*Create an archive file*

1) In the Applications view, right-click the application version and select Create AEM Forms Archive.
2) Select Create A New Archive File and click Next.

3) Select applications and/or assets to be included in the archive file. The following rules apply when selecting applications and assets:
   - To make a selection, select the check box beside a component. To deselect a component, select the check box again.
   - A shaded box indicates that child components are selected implicitly.
   - Selecting an application version automatically selects all assets within the application.
   - Selecting a folder automatically selects all files and folders contained within it.
4) In the Local File System Destination box, specify the file name and location of the archive:

   **NOTE:** You do not need to specify the file name extension .lca. It will be automatically added to the file name.
5) (Optional) In the Archive Description box, type a description for the archive so that others will know what its contents are. A default description specifies the date and time of the archive creation and the name of the user who created the archive.
6) Click Finish to save the file.

The archive file is saved in the specified location. You can use administration console to import it into another server. (See "Managing AEM Forms Archives" in AEM Forms administrationhelp

.)

*Create an archive file based on existing file*

1) In the Applications view, right-click the application version and select Create AEM Forms Archive.
2) Select Create A New Archive File From A List Of Assets In Existing Archive File.
3) In the Archive File box, type a full path to the archive file you want to start with or click the ellipsis button and select the file.
4) Click Next and select applications and/or assets to add to the archive file or deselect applications and/or assets to remove.
5) In the Local File System Destination box, specify the name of the archive file to create.
6) (Optional) In the Archive Description box, type a description for the archive so that others will know what its contents are. The default description includes information about the archive creation date and the name of the user who created the file.
7) Click Finish to save the archive file.

*Create a patch for an existing archive*

1) In the Applications view, right-click the application version and select Create AEM Forms Archive.
2) Select Create A Patch for An Existing Archive and click Next.
3) Select the assets that need to be updated.
4) In the Local File System Destination box, specify the name of the archive file to create.
5) (Optional) In the Archive Description box, type a description for the archive. The default description includes information about the archive creation date and the name of the user who created the file.
6) Click Finish to save the archive file.

*About runtime configuration XML files*

Each application and application version has runtime configuration settings associated with it. A runtime configuration settings file is an XML file that contains information about applications, such as:

- Service configuration
- Pool configuration
- Endpoints details
- Security profile

To view or modify the runtime configuration settings of an application, export the settings to a runtime configuration XML file. You can use an XML or text editor to open or modify the contents of the runtime configuration XML file. When you want to modify runtime configuration settings of an application, import the modified XML file using administration console.

For example, when you move an application to a new environment using a AEM Forms archive file, you may need to modify the runtime configuration settings to suit the target environment. AEM Forms archive files are in a compressed format, and can be read only by the AEM Forms server. Therefore, you need a runtime configuration settings XML file to view and edit application settings.

Export runtime configuration settings

You can export configuration settings as an XML for an application or application version.

1) In the Applications view, right-click an application or an application version and select Create AEM Forms Archive.
2) Select Export Runtime Configurations.
3) In the Export Runtime Configurations File box, type the full path and name of the configuration file you want to create.
4) Click Finish to save the runtime configuration file.

For information on importing runtime configuration files, see "Import and manage AEM Forms applications and archives" in AEM Forms administrationhelp

.

## Avoiding socket time-outs when creating archives

Socket time-outs can occur when you add many or large resources and processes to an archive. Create archives asynchronously if socket time-outs occur when you are creating or updating archives.

When you create archives, Workbench opens a connection to the AEM Forms server. The default behavior is to maintain the connection while the archive is created. If the connection is open for too long, a socket time-out can occur. The amount of time required to cause a socket time-out depends on the server configuration.

When you are creating archives asynchronously, the connection is closed after Workbench sends the request to the AEM Forms server to create the archive. Another connection is opened when the AEM Forms server notifies Workbench that the archive is created.

**Create archives asynchronously**

1) Open the workbench.ini file in a text editor.

   *The workbench.ini file is in the [install directory]/AEM Forms Workbench/workbench directory, where [install directory] is the location where you installed Workbench.*

2) Locate the following line of text that controls the asynchronous behavior of archive creation:

   `-Dcom.adobe.workbench.repository.create_lca_asynchronously=false`

3) Change the text to set the property to true:

   `-Dcom.adobe.workbench.repository.create_lca_asynchronously=true`

4) Save workbench.ini and restart Workbench.

# 4.4. Creating Forms

Use the Form Design perspective to create form designs for AEM Forms applications. Create the form in Workbench and design the form in Designer. When you open Designer from Workbench, you can also create form designs in Designer and then check them into Workbench.

You can create form designs for automated business processes that require user interaction. For example, an online mortgage application starts when a customer fills an online application form and submits it to the bank. The bank then reviews the application and sends the status of the application to the customer.

The Form Design perspective is a collection of default views for creating form designs. It includes the views that you use to create a form design in Designer:

**Applications:**

A container for storing assets that are required for implementing a AEM Forms solution. Examples of assets are form designs, fragments, images, processes, DDX files, JavaServer pages, HTML pages, and SWF files

**Resources:**

A hierarchical display of all the content in the repository. Provides access to the folders that contain form designs, images, fragments, schemas, and WSDL, XML, and SWF files.

**Editor:**

Displays an image of the first page in a form design. The editor opens automatically when you open a form. Tabs in the editor area indicate the names of the forms that are currently open for editing. An asterisk (*) beside the name indicates that the file or object has unsaved changes.

You can switch between the Form Design perspective and other perspectives while you work. As you become more familiar with Workbench, you can open views that the Form Design perspective does not contain. (See Perspectives.)

Designer does not communicate with the AEM Forms repository. If your form references assets in a AEM Forms application, synchronize those files on your local file system. If you do not synchronize the referenced files, they are unavailable to use in Designer.

## Create a form design using Workbench:

1) Open the Form Design perspective if it is not already open. (See Opening the Form Design perspective.)
2) Create an application and add resources. (See Organizingyour forms and assets.)
3) Create the form design. (See Creatingthe form design.)
4) Save the form design. (See Savingthe form design.)
5) Preview and test the form design:
   – Preview your form design as you work by using the Preview PDF tab in Designer. When you preview, Designer renders your form as a PDF file.
   – You can test a form by using sample data. Using Designer, you can automatically generate sample data to preview and test your form. (See “To automatically generate sample data to preview your form” in Designer Help
   – .)
6) Prepare the form design to work with AEM Forms - forms workflow.

   *Form designs that become part of a process require certain objects to function. An example is a form that is part of an application that you want to make available in Workspace. Add a button object to form designs for them to work in Workspace. The button object must be a submit button or a button that references a submit button.*

   *Designer includes two custom objects named Process Fields and Form Bridge that you can use to make forms work in Workspace. For more information about preparing a form design for forms workflow, see “Creating forms for forms workflow” in Designer Help*

   .

## Opening the Form Design perspective

Open the Form Design perspective to create form designs.

The Form Design perspective is associated with form designs, which you can save in XDP and PDF. When you create or open a form design, the file opens in Designer, which operates outside Workbench. Workbench displays a tab for each form design that is open in Designer. The tab displays an image of the first page in the form design. Clicking the image on a tab automatically switches to the open form in Designer.

**Open the Form Design perspective:**

1) Do one of the following actions:
   – In the toolbar, click the Open Perspective button , and then select Form Design.
   – Click Window > Open Perspective > Other, select Form Design, and then click OK.
   – Click Window > Open Perspective > Form Design.

   *By default, perspectives always open in the same window. To open perspectives in a new window, edit the preferences.*

## Organizing your forms and assets

In Workbench, use the Applications view to create an application to store your form designs and assets, such as images and fragments. After you create an application, create folders to organize your form designs and assets according to your needs. For example, you many want to keep your form designs in the top folder and place fragments and images in subfolders. In this example, place fragments in a subfolder named *Fragments* and image files in a subfolder named *Images*.

After you create forms or assets, check them into Workbench so that they are available to other developers.

Before you edit a form design, synchronize the assets on your local file system. If you do not synchronize the files, they are unavailable for use in Designer.

## Creating the form design

With the New Form wizard, you can create XDP and PDF form designs. The New Form wizard starts in Workbench and continues in Designer.

The New Form wizard guides you through a series of steps where you choose the type of form design to create, how people fill it, and how the information is submitted. Choose one of the following methods to create a form design:

**Use a Blank Form:**

Allows you to specify the page size, orientation, and number of pages.

**Based on a template:**

Allows you to select a predefined template to base the form design on. The templates are divided into several categories.

**Import a PDF document:**

Allows you to import a PDF file from the local file system. You can specify if you want to import the PDF file as an interactive form with either fixed pages or a flowable layout. If you select a data model, the option to create an interactive form with fixed pages is unavailable because the form is saved as a PDF file.

**Import a Word document:**

Allows you to import a Word document from the local file system.

When you finish the New Form wizard, the form opens in Designer, where you can design the form. Workbench displays a corresponding tab for each form design that is open in Designer. The tab displays an image of the first page in the form design. When you click the image on a tab, you switch to the open form in Designer.

**Create a form design:**

1) Open the Form Design perspective if it is not already open. (See Openingthe Form Design perspective.)
2) Do one of the following actions:
   – Click File > New > Form.
   – Click File > New > Other > AEM Forms > Form, and then click Next.
   – In the Applications view, right-click an application version and then click New > Form.

     **NOTE:** This option is not available when you select the root application folder.
3) Follow the on-screen instructions to create a form.

**New Form**

The New Form screen is the first screen in the New Form wizard. On the New Form screen, specify the form name and the location to create the form design.

**NOTE:** Clicking the Finish button creates an XDP form using the default settings in the remaining New Form wizard screens.

**Name:**

The name of the form design.

**Description:**

(Optional) A description of the form design. The description appears in the Properties view. It also appears in Designer, in the Form Properties dialog box.

**Location:**

A list of all the applications that are synchronized on the local computer. Select the folder where you want to create the form design.

**New Folder:**

Creates a folder in the selected location. Specify the name of the folder.

**Specify Form Data Model**

On the Specify Form Data Model screen in the New Form wizard, specify a data model for the form.

Selecting a data model creates an XDP form that is a document of record. You can create only a non-interactive PDF form. When you click the Next button, the Opening Designer screen appears.

**NOTE:** Clicking the Finish button creates an XDP form using the default settings in the remaining New Form screens.

**Specify a data model for this form (data model or schema):**

You can select or import a data model or an XML schema, or select no data model. Different options appear based on your selection.

**Select a data model from a AEM Forms application:**

Select an XML schema or a data model from an application on your local computer.

**Import a data model:**

Import an XML schema or a data model file from your local computer to the current application.

**No data model:**

(Default) If you select No Data Model, the other options are disabled. If necessary, you can add a data model or schema in Designer.

**Enter or select a data model:**

This option is available only when you select Select A Data Model From A AEM Forms Application. Select the data model for your form. The folder list displays all applications that are synchronized on your local computer. The list displays only folders, schema files (XSDs), and data model files (FMLs).

**Import data model:**

This option appears only when you select Import A Data Model. Select the data model file to import.

**Enter or select the parent folder:**

This option appears only when you select Import A Data Model. Select the parent application folder where you want to store the data model file.

**New Folder:**

This button appears only when you select Import A Data Model. Click this button to create a folder for the imported data model.

### Form Usage

On the Form Usage screen in the New Form wizard, specify how the form filler submits the form. You can also configure Acrobat and Adobe® Reader® features for the form.

**NOTE:** Clicking the Finish button creates an XDP form using the default settings in the remaining New Form screens.

**Form Submission**

Specifies how the form filler submits the form. You can also add a Submit button later in Designer. Form designs that become part of a process require certain objects to function. An example is a form that is part of an application that you want to make available in Workspace. Add a button object to form designs for them to work in Workspace. The button object must be a submit button or a button that references a submit button.

**From AEM form workspace:**

(Default) The form filler submits the form from Workspace. A Form Bridge custom object and a Submit button are automatically added to the form design.

**Through email:**

The form filler submits the form through email. A Process Fields custom object is automatically added to the form design. The Process Fields custom object includes a Submit button.

*Adobe Acrobat/Reader*

Configure the following options if the form filler must use Acrobat or Adobe Reader to submit the form:

**Submit using Adobe Reader 9.1 or later:**

The form filler must use Adobe Reader 9.1 or later to submit the form. A Submit button is not added to the form design. Select the format from the Submit Adobe Acrobat/Reader Form As option.

**Features that Adobe Reader users will use:**

Specifies features that form fillers use in Adobe Reader.

**Commenting:**

Enables commenting on the form and creates a fixed form. If you create an XDP form, selecting this option sets the default server rendering and the PDF preview formats to fixed. Use the Reader Extensions service to apply usage rights.

**Digital signatures or certification:**

Enables digital signatures or certification on the form. The form submit type must be PDF and the Submit Adobe/Reader Form As option is set to PDF. If you also select Use Adobe Reader 9.1 Or Later, a Submit button is not added to the form. Use the Reader Extensions service to apply usage rights.

**Encryption:**

Enables encryption on the form.

**Submit Adobe Acrobat/Reader form as:**

Specifies the submission format of the form. The default is XDP. However, if you select Digital Signatures Or Certification, it is set to PDF. If the submission format is PDF, use the Reader Extensions service to apply usage rights.

**Opening Designer**

The Opening Designer screen indicates that you continue the New Form wizard in Designer. In Designer, the Getting Started screen appears, where you select a method for creating the form.

## Opening form designs

When you open a form design in Workbench, the form design opens in Designer. In Workbench, a tab displays an image of the first page of the form design. Clicking the image switches to the open form design in Designer, where you can edit the form.

You can open multiple form designs in Workbench, which displays each form design in a separate tab.

Closing a form design in Designer closes the corresponding tab in Workbench. Closing a tab in Workbench closes the corresponding form design in Designer.

**Open a form design:**

1) Do one of the following actions:
   – In the Applications view, double-click the form design. The form design opens in Designer. A corresponding tab appears in Workbench and displays an image of the first page of the form design.
   – If the form design is already open, click the corresponding tab and then click the image of the form design. The focus switches to the open form design in Designer.

**Close a form design:**

1) In Designer, save the form design.
2) Close the form in Designer or in Workbench:
   – In Designer, close the form design. The corresponding tab in Workbench closes.
   – In Workbench, close the tab. The corresponding form design closes in Designer.

## Saving the form design

Save your form design in Designer. After you save a form, check it into Workbench to ensure that it is available to others who are connected to the same server.

When saving PDF files, you can select whether you want to save the file as a Dynamic XML form or a Static PDF form. (See “Save Options (Form Properties dialog box)” in Designer Help

.)

**NOTE:** When the connection to the server is interrupted while you are working in Designer, the changes you make to a form are not saved. A message notifies you that the connection has been interrupted but no warning message appears indicating that the form was not saved. When changes are not saved, an asterisk (*) appears on the tab form name and the version does not increment in the Resources view. To avoid losing your changes, save the file locally and then reimport it in Workbench later.

## Where to find more information

If you are new to Designer, you can access the topics in Designer Help

and Designer Quick Starts

.

Designer includes a selection of complete sample forms. Each one includes a form design, sample data or schema, and the final version of the form. The sample forms illustrate both simple and complex form design techniques. The sample forms are installed in the EN\Samples folder under the Designer installation folder.

Designer Help

is available from the Help menu when you open a form design. For more information about forms, see the following topics in Designer Help

:

- Getting Started
- Sample Forms
- Using Designer > Working with Form Designs > Creating, opening, and saving > Opening and saving forms > To select the Acrobat and Adobe Reader target version
- Working with Form Designs> Importing Documents > Importing PDF documents as artwork
- Using Designer> Working with Data Sources > Connecting to a data source > To create a data connection to an XML schema and XML data file
- Working with Form Designs > Creating Forms for Process Management 11.

## Creating XML Schemas

An XML Schema is a description of a type of XML document. Typically, the description is expressed in terms of constraints on the structure and content of the XML document.

You can create an XML Schema Definition file (XSD), connect it to a form, and bind specific elements and attributes defined in the schema to fields in a form design. You use this to map data into and out of form fields in a format that conforms to the schema.

The procedures in this section presume you understand how to create and edit XML schema.

### Create an XML schema

Use the XML Schema Editor to create XML schema.

1) Click File > New > XML Schema. The New XML Schema dialog appears.
2) In the Name box, type a name for the schema.
3) (Optional) In the Description box, type a description for the schema.
4) Enter or select a parent folder location for the XML schema. The schema appears in the Application view.
5) Click Finish. The XML Editor appears.
6) Create the schema as needed. For example, add elements and attributes to define your XML Schema Definition file. You can use the Design view or Source view to accomplish this task.
7) To save the schema, do one of the following:
   - To save the schema in the previously specified location, click Save.
   - To save the schema in another location, click Save As, in the Save As dialog box specify a parent folder location and file name for the schema, and click OK.

### Edit an XML schema

Use the XML Schema Editor to edit XML schema.

1) In the Application view, right-click on the schema to edit.
2) Click Open. The schema opens in the XML Schema Editor.

3) Make changes as needed.
4) To save the schema, do one of the following:
   – To save the schema in the previously specified location, click Save.
   – To save the schema in another location, click Save As, in the Save As dialog box specify a parent folder location and file name for the schema, and click OK.

RELATED LINKS:

Working with Applications

Developing applications in a multi-developer environment
XML Schema Tutorial

Introduction to the XSD Editor

# 4.5. Managing Resources

In Workbench, resources are files that are not part of the applications, or that are programmatically delivered directly into the AEM Forms repository. An example of resource is content migrated from Live-Cycle.

**NOTE:** To view resources that are directly associated with an application, see the Applications view.

The Resources view is not displayed by default in the Process Design perspective and the Form Design perspective. Select Window > Show View > Resources to open it. The view contains a hierarchical list of the resources in the repository. In the Resources view, you can browse the resources or do the following actions:

- Display a filtered list of resources.
- View dependencies between resources.
- View the version history.
- Set access permissions on resources.

## Filtering resources

The Resources view initially shows all content in the repository that you have permission to view. You can further restrict what is displayed by filtering the resources.

**Filter the resources in the Resources view:**

1) Click one of the Resources view toolbar buttons:

**Filter for PDF Files :**

Only resources with the filename extension .pdf are displayed. (See Ensuringthat the PDF icon is displayed.)

**Filter for Image Resource :**

Only image resources are displayed. Image resources have the filename extension .png, .gif, .jpeg, and so on.

**Filter for XDP Form :**

Only resources with the filename extension .xdp are displayed.

**Filter Resources :**

All resources are displayed.

**Ensuring that the PDF icon is displayed**

PDF files may initially be displayed with the icons for XDP files in the Resources view. Do this procedure to display the Adobe Acrobat PDF icon instead.

*Display the PDF icon for PDF files:*

1) Select Window > Preferences.
2) In the Preferences dialog box, select General > Editors > File Associations.
3) In the File Types box, select *.pdf.
4) Beside the Associated Editors box, click Add.
5) In the Editor Selection dialog box, click External Programs, select Adobe Acrobat Document, and then click OK.
6) Beside the Associated Editors box, click Default, and then click OK.
7) In the Resources view, click the Refresh Selection button.

## Viewing resource relationships

Resources listed in the Resource view can be reused by other resources. While this method improves the consistency and efficiency of the application design, it also creates dependencies between these assets. Thus, before you modify or delete a shared resource, check how this change will affect other resources. You can examine resource relationships in the Relationships dialog box.

The Relationships dialog box displays a table of all the resources that are referencing the selected resource. The table contains the following information for each resource:

**Name:**

The path to the resource. It includes the filename of the resource.

**Date:**

The date the latest revision was created.

**Version:**

The current version of the resource.

**Created By:**

The username of the user who originally created the resource.

**Last Modified By:**

The username of the user who created the latest version of the resource.

**View the resource relationships:**

1) In the Resources view, right-click the resource and select Relationships.

**Sort the list:**

By default, the list is sorted by the Name column in ascending order. You can sort the list by any other column.

1) Click the column heading to sort the list by that column.
2) (Optional) Click the column heading again to change the sorting order.

## Working with file versions

The version number of a file is updated automatically when a modified file is saved. You can display a history of all versions of a file, and you can promote an older version to be the current version. When a resource is referenced, its current version is used.

The version number is displayed in brackets beside the filename. The version number is of the form #x.y, where x is the major version and y is the minor version. When a file is first created or added to the repository, its version is #1.0.

The major version increments only when explicitly requested. (See Usingolder versions of files.) The minor version increments each time the file is changed and saved or is replaced.

### Viewing the version history

You can view the history of a file, including the version number, date of last modification, user who change it, and comments that are associated with the version.

You can also save any version of a resource with another name or to another location. This method allows you to review a previous version without making it the current version.

*View the version history of a file:*

1) Right-click the file and select History. The Versions dialog box appears.

*Save a version of a resource to another location:*

1) In the Versions dialog box, select a version and click Save As.
2) In the Save As dialog box, specify a location and filename, and then click Save.

### Using older versions of files

You can promote an older version of a file to be the current version. This procedure provides a means to retrieve an older version of a resource that has been overwritten by subsequent modifications. Promoting it to the current version causes a new version to be created with the content of the specified older version. You also have the option of incrementing the resource's major version number to signify major changes to the resource. You may also need to do this if, for example, the latest version of a file contains an error and you want to revert to a previous version.

*Promote an older version of a file to be the current version:*

1) Right-click the file and select History.
2) Select the version of the file and click Promote to Current.
3) Type a comment to explain why the version is being promoted.
4) (Optional) Click Update Major Version. If this option is selected, the major version (the number before the dot) is incremented and the minor version (the number after the dot) changes to zero. If this option is not selected, only the minor version is incremented.
5) Click OK, and then click Close.

The promoted file is now displayed in the Resources view with its new version number, and this version of the file is used by any service that refers to it.

## Setting access permissions

If you have sufficient permissions, you can use the Resources view to view, add, or remove access permissions to the resources in the repository. Typically, you have sufficient permissions for these tasks if you are logged on as an administrator or if an administrator assigned you the required permissions.

### About access permissions

When permissions are added to a folder, they apply to a specific user or group. The following permissions are available:

**Delegate:**

Users can assign access permissions to other users of the folder.

**Read:**

Users can open the resources in the folder.

**Traverse:**

Users can view the folder’s contents and can navigate through the folder to lower-level folders. This permission is used when a user needs Read or Write permissions for a lower-level folder but not for the resources in higher-level folders.

**Write:**

Users can modify and save the resources in the folder.

Users who are assigned the Super Administrator or Application Administrator roles have full access to all resources, regardless of any access permissions. User roles are administered using administration console. For all other user roles, permissions must be explicitly added to the folders.

Permissions are checked when a user initiates an action that requires support from the repository. (See also, AEM Forms administration help

.)

### Viewing and changing access permissions

You can view and change the current permissions that users or groups have for a folder.

By default, changes to folder permissions are applied to all resources in the folder and to the resources in subfolders. However, you can specify that the changes apply only to the resources in the folder.

1) Right-click the folder and select Access Control.

   *If you do not have sufficient permissions, the Access Control command is dimmed.*

2) Select the user or group that you want to change access permissions for. The access permissions of the user or group for that folder are displayed in the Permissions area.
3) Select the permissions that you want to add, and deselect the permissions that you want to remove.
4) Specify whether to apply the changes to subfolders:
   - To apply changes to the resources in the folder as well as the resources in all subfolders, select This Folder And All Resources And Folders It Contains.
   - To apply changes only to the resources in the folder, select This Folder And Only The Resources It Contains.
5) Click OK.

### Adding access permissions

Add access permissions to configure access to resources for users or groups. Permissions are added at the folder level. All resources in a folder have the same access permissions.

By default, folder permissions are applied to all resources in the folder and to the resources in subfolders. However, you can specify that the changes apply only to the resources in the folder:

- Add permissions to a folder and subfolders when users need the same access permissions for them.
- Add permissions only to the folder when users need different access permissions for different folders.

For more information about how to manage access permissions that are different for subfolders, see Addingdifferent permissions to different subfolders.

*Add access permissions:*

1) Right-click the folder and select Access Control.

   *If you do not have sufficient permissions, the Access Control command is dimmed.*

2) To specify the users and groups to provide access permissions for, click Add. The Add Users and Groups dialog box appears.
3) Search for users and groups:
   – In the Search Type area, select User Name or Email to specify the attribute to search on.
   – In the box, type the text that you want to match with the user attribute. (The search is not case sensitive.) Click Find.
4) In the Results box, select the users and groups to add and click Add.
5) Repeat steps 3 and 4 until the Selected User(s) Or Group(s) box contains all the users and groups that you want to configure access permissions for, and then click OK.
6) In the Users And Groups box of the Access Control dialog box, select one or more users and groups to configure access permissions for.
7) In the Permissions area, select the permissions to provide to the selected users and groups.

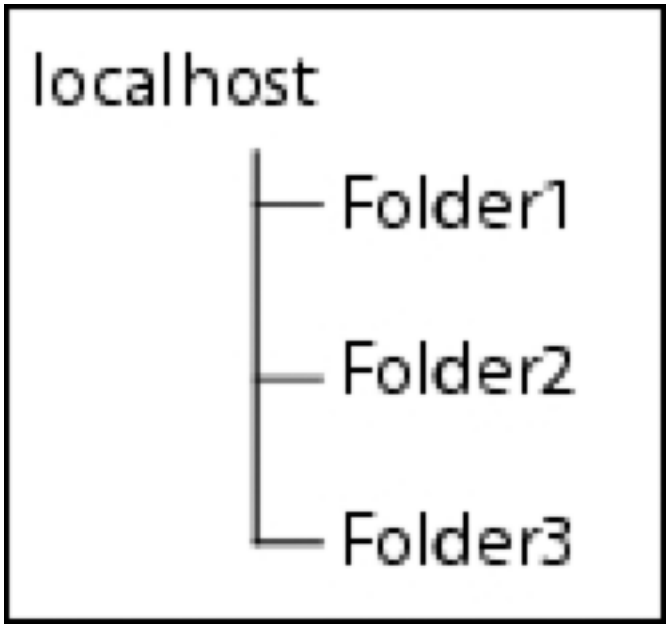
   *If you select the Read permission, Traverse is automatically selected. If you select the Write permission, both Read and Traverse are automatically selected.*
8) Repeat steps 6 and 7 until you have configured access permissions for all the users or groups.
9) In the Propagate Permission Changes area, select an option to specify whether the access permissions are applied to the folder as well as the subfolders, or only to the folder.
10) Click OK.

**Adding different permissions to different subfolders**

When you add access permissions to resources, you can add them only to a folder, or to the folder and all its subfolders. To quickly implement different access permissions for different subfolders, use the following strategy:

- For all users, add permissions to top-level folders that all or most of the users require. Specify that the permissions are propagated to all the subfolders as well.
- Then, for subfolders, add or remove permissions for specific users.

For example, three folders, Folder1, Folder2, and Folder3, are created under the root named localhost in the Resources view.




Permissions must be added for two user groups, GroupA and GroupB:

- GroupA can see and access only Folder1.
- GroupB can see and access only Folder2.
- Neither group can see or access Folder3.

The following procedure is used to implement the access permissions:

1) Use administration console to create a user group (ResourceUsers) that includes GroupA and GroupB.
2) Add Traverse permissions for the ResourceUsers group to the root (localhost), and specify that the permissions propagate to all subfolders. This step enables all users to navigate the entire folder tree.
3) Add Read, Write, and Traverse permissions for GroupA to Folder1.
4) Add Read, Write, and Traverse permissions for GroupB to Folder2.
5) Remove all permissions of the ResourceUsers group from Folder1, Folder2, and Folder3.

   **TIP:** When additional users must access a folder, they can be added to either GroupA or GroupB.

**Removing access permissions**

Remove access permissions if users or groups do not need access to resources in a folder.

*Remove access permissions:*

1) Right-click the folder and select Access Control.

   *If you do not have sufficient permissions, the Access Control command is dimmed.*
2) Select the users or groups to remove permissions from and click Remove.
3) In the Propagate Permission Changes area, select an option to specify whether the access permissions are removed from the folder as well as the subfolders, or only from the folder.

4) Confirm your selection and click OK.

# 4.6. Managing Components and Services

An administrator can manage components and services in the Components view. A component is a collection of services delivered as a JAR file. The state of the components and services determines which services and operations are available to Workbench users in the Services view.

To be made available in the Services view, components and services must be installed, started, and activated in the Components view. Depending on how a component was developed, the installation step may automatically perform the start and activation steps.

Components can also be started and stopped by an administrator in administration console.

## Opening the Components view

The Components view is not displayed by default because it is intended for use by administrators or component developers.

1) Select Window > Show View > Components.

   *You must have sufficient permissions to view and manage components. If no components are displayed, see your administrator.*

## Installing components

You can install additional components you have developed in the Components view.

Upon installation, most components are started automatically and their services are activated automatically, which makes them available immediately to users in the Services view. If the component does not start automatically, you can start it as described in Startingcomponents and services.

**To install a component:**

1) Open the Components view.
2) Right-click the Components folder and select Install Component.
3) Navigate to and select the component JAR file from your local computer or from a network location.

## Patching components

You can patch a component if you are supplied with a JAR file specifically for that purpose.

Patching a component allows a component to be replaced with an updated version while retaining any system data that the component has accumulated to support existing running processes.

**To patch a component:**

1) In the Components view, stop the component to be patched. (See Stopping componentsand services.)
2) Right-click the stopped component and select Patch Component.
3) Navigate to and select the patch JAR file from your local computer or from a network location, and then click Open.
4) If necessary, restart the component. (See Startingcomponents and services.)

## Starting components and services

Upon installation, most components and their services are started and activated automatically, which makes them available in the Services view. If the component does not start automatically when it is installed, you can start it manually.

A component may not start automatically if, for example, its parameters need to be configured or if the service has some kind of restriction on its availability.

Components can also be started in administration console.

**To start a component:**

1) In the Components view, navigate to the component you want to start.
2) Right-click the component and select Start Component.

   **TIP:** You can select multiple components to be started at once by Ctrl-clicking.

**To start a service:**

1) In the Components view, navigate to the service you want to start. Its component must already be started.
2) Right-click the service and select Start Service.

   *After you start a service, it also needs to be activated before it can be used. If you need to activate a service manually, follow the steps in Activatingservices.*

## Activating services

To use a service in a process, the service must be active. Most services are activated automatically when their component is started; therefore, doing this step manually is usually not required. When a service is activated, it is available for use in the Services view.

**To activate a service:**

1) In the Components view, right-click the service and select Activate.

## Editing service configurations

If a component exposes service configuration values, you can edit those values in the Component view. These values depend entirely on the implementation of the component and its services.

**TIP:** Process services are grouped in the WorkflowDSC component. You can use this procedure to set the values of configuration parameters that are defined in a process.

**IMPORTANT:** In the Designtime Service, the value for the Application Root parameter should not be removed after the service is installed.

**To edit a service configuration:**

1) In the Components view, expand the component that provides the service that you want to configure.
2) Right-click the service and select Edit Service Configuration.
3) In the dialog box that appears, edit the values and click OK.

   *If a component does not expose any configuration values, the menu item is dimmed.*

## Removing service configurations

After a service has been deactivated, you can completely remove that instance of the service, thereby preventing it from being activated again.

**IMPORTANT:** The active services that are listed for the WorkflowDSC component are services that are created for process versions. If you remove the service configuration for a process version, the process version is deleted.

**To remove a service configuration:**

1) In the Components view, right-click a deactivated service and select Remove Service Configuration.

## Stopping components and services

You can stop a component or service for maintenance, such as changing the configuration parameters or installing a new version, or to prevent it from being used in process designs.

**To stop a component or service:**

1) In the Components view, navigate to the component or service you want to stop.
2) Right-click and select either Stop Component or Stop Service.

   **TIP:** Administrators can also use administration console to stop components.

### Deactivating services

You can deactivate an individual service within a component if you do not want that service to be used.

**To deactivate a service:**

1) In the Components view, right-click the service and select Deactivate.

### Uninstalling components

If a component is not used in any process, you can remove it from the AEM Forms Server by uninstalling it.

Uninstalling a component removes it from the server but does not affect the component's originating JAR file. The JAR file is located on a local computer or network location.

**To uninstall a component:**

1) Right-click the component and select Stop Component.
2) Right-click the component and select Uninstall Component.

## 4.7. Customizing Perspectives

You can customize or personalize a perspective so that the layout suits your individual preferences. For example, you can change the location of the views in a perspective and then save the perspective so that the views are always arranged the same way. You can also restore a perspective to its original default layout.

### Moving views

If you prefer to have a view in a different location in Workbench, you can move it to almost any other location.

1) Drag the view by its title bar. As you drag the view within or outside of Workbench, the mouse pointer changes to one of these docking options to indicate where the view will be docked when you release the mouse button:
   - **Dock above**: The view is docked above the view that is below the pointer.
   - **Dock below**: The view is docked below the view that is below the pointer.
   - **Dock to the right**: The view is docked to the right of the view that is below the pointer.
   - **Dock to the left**: The view is docked to the left of the view that is below the pointer.
   - **Stack**: The view is docked as a tab in the same pane as the view that is below the pointer.
   - **Restricted**: You cannot dock the view in this area.
2) When the view is in the location that you want, release the mouse button.

**TIP:** You can also move a view by using the context menu for the view, which is displayed when you right-click the view's tab header.

## Saving perspectives

If you modified a perspective by moving its views, adding or removing views, or by making other similar changes, you can save these changes for future use.

1) Switch to the perspective that you want to save.
2) Select Window > Save Perspective As.
3) In the Name box, type a new name for the perspective, and then click OK.

## Restoring perspectives

If you have changed a perspective but are not satisfied with the layout, you can restore (reset) it to the original layout.

1) Select Window > Reset Perspective and click OK.

# 5. Getting started with process design

## 5.1. About Processes

Processes represent the business processes that you are automating using AEM Forms. You use Workbench to model business processes at design time. At run time, processes are services that run on the AEM Forms Server. Client software initiates processes on demand according to business needs and the purposes of the processes.

### Opening the Process Design perspective

Open the Process Design perspective to create or edit process diagrams.

The Process Design perspective is composed of an editor and several views. The editor is a visual component that is used to edit or browse a process diagram. The views support the editor, provide alternative presentations for selected items in the editor, and let you navigate the information in Workbench.

**To open the Process Design perspective:**

1) Click the Process Design Perspective icon in the toolbar.

   **TIP:** You can also click the Window menu and select either of the following items:

   – Open Perspective > Process Design
   – Open Perspective > Other > and then select Process Design and click OK.

Window > Open Perspective > Process Design is available only when the Process Design or Form Design perspective is currently being used.

By default, perspectives always open in the same window. To open perspectives in a new window, edit the preferences.

### Process Design perspective views

The Process Design perspective contains a default combination of views and an editor. You can also open views that the current perspective does not contain.

These views are available:

**Components:**

Tools for installing and managing components.

**Events:**

Tools for creating and managing event types. (See Managingevent types.)

**Applications:**

Tools for viewing and managing AEM Forms applications.

**Process Properties:**

Tools for configuring the properties of operations on a process diagram. (See Inputand output data for operations.)

**Resources:**

Tools for interacting with resources in the repository, such as folders, forms, and images.

**Services:**

Lists the services you can use in your process diagrams.

**Validation Report:**

Tools for viewing the results of process validations. (See Validationreports.)

**Variables:**

Tools for creating and configuring variables. (See Processvariables.)

After you open a view, it becomes part of the current perspective until you close the view.

**To open a view:**

1) Select Window > Show View > *view*, where *view* is the name of the view.

RELATED LINKS:

Opening the Process Design perspective

# 5.2. Services

*Services* are components that run on the AEM Forms Server that execute operations when requested by client software. Each service provides one or more operations that you use to represent activities in processes.

For example, the LDAP service provides the LDAP Query operation. When your process needs to retrieve information from an LDAP directory, you use this operation.

The Services view in Workbench lists the services that are available, organized in categories. The services that are available depend on the following factors:

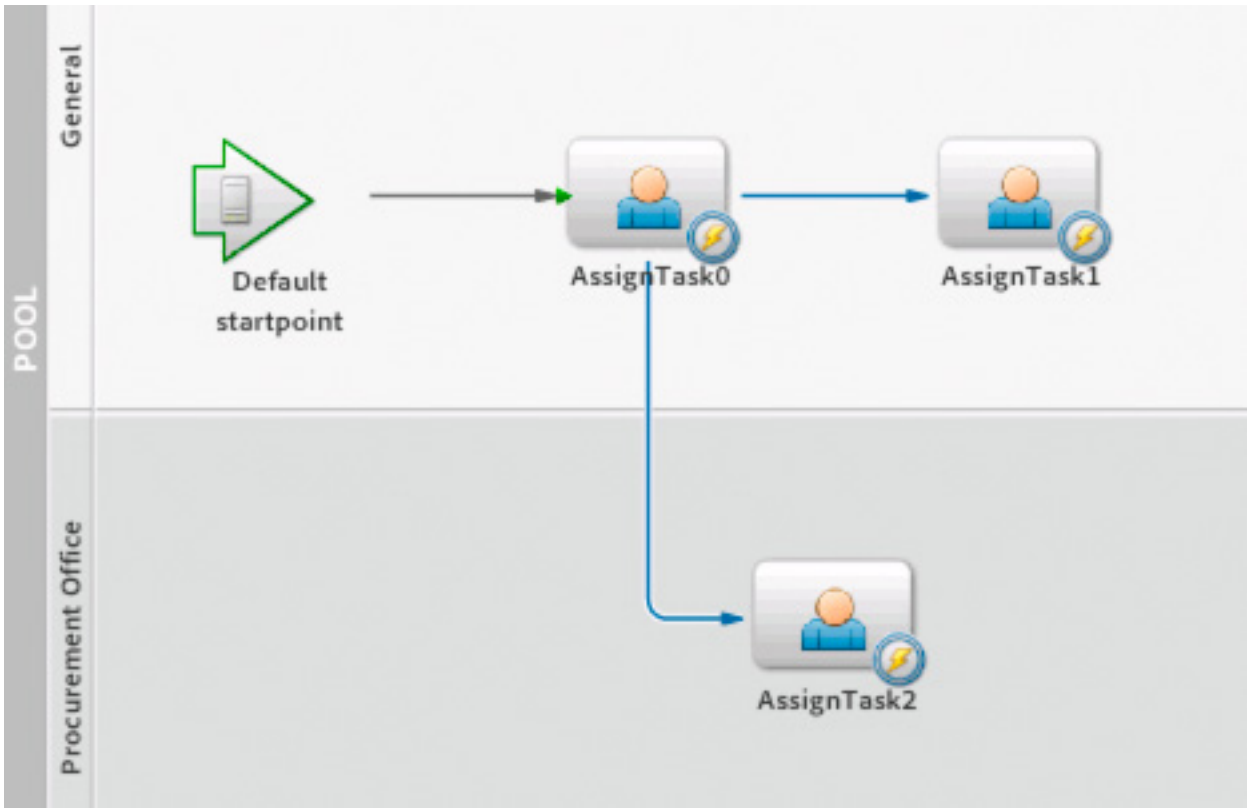
- The services have been deployed. You can use the Components view to deploy services.
- Most services are provided by AEM Forms solution components.
- Some services are included with Foundation.
- Processes that are activated are available to use as services. You can use these services as *subprocesses* within other processes.

RELATED LINKS:

Service reference

## 5.3. Process diagrams

*Process diagrams* are visual representations of the business process that is being automated. Process diagrams are created at design time to represent the order in which the activities in a process are executed. Process diagrams also indicate where decisions need to be made to determine the next activity to perform when several possibilities are available.




Process diagrams are displayed in the editor that is included in the Process Design perspective. You can add the following types of elements to process diagrams:

**Operations:**

The steps on the process that represent the execution of service operations.

**Routes:**

The arrows that join operations and denote the order of execution.

**Swimlanes:**

Rows that provide a visual organization of operations.

**Events:**

The notification or reaction to the occurrence of an event.

**Gateways:**

Parallel branches that enable operations to execute simultaneously.

**Abstract activities:**

Placeholder steps that represent undefined service operations.

RELATED LINKS:

Process data

Working with operations

Drawing routes to link operations

Organizing operations in swimlanes

Controlling process flow using events

# 5.4. Process data

Processes typically require data to function. For example, data can be introduced to the process as input when the process is initiated or at other activities in the process, and can be used during the process as required. Data can also be provided as output and returned to the process initiator when the process is complete. Processes store data in variables that can be referenced and used for making decisions, or acted on by services operations.

Variables can be used in two ways:

- Process variables are used to store and access data at run time. The data stored is expected to be different for each process instance, and can be changed at run time. (See Processvariables.)
- Configuration parameters are used to set default data values that apply to all process instances. The data cannot be changed at run time, but can be configured using administration console or Workbench. (See Configuration parameters.)

## Process input and output data

You use process variables to specify data that can be provided as input when processes are initiated. Similarly, process variables can be used to store output data that is returned to the process initiator when the process is complete. You can also make it mandatory to provide input data when a process is initiated.

The values of the General group of variable properties determine whether a variable stores process input and output data. (See Commonvariable properties.)

## Process data model

Data that is saved for a process is represented internally as a data tree. Each node in the data tree represents a data item that is available to the process. The root node of the tree is named `process_data`. Below this node are all other data nodes. The following standard nodes are present for all processes:

**create_time:**

The date and time when the process instance was created.

**creator_id:**

The identification of the user who initiated the process instance.

**id:**

The unique identifier for the process instance.

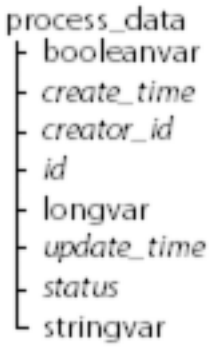
**status:**

The life cycle status of the process instance.

**update_time:**

The date and time when the process instance data last changed.

Any other nodes below `process_data` represent data stored in process variables that have been created for the process. The following graphic illustrates the data tree for a process that has three variables defined and named `booleanvar`, `stringvar`, and `longvar`.




Variables appear in alphabetical order in the data tree. The names of the standard nodes are displayed using italics.

Simple variable types, such as `boolean` and `long`, are represented by leaf nodes in the data tree. Some variable types, such as `xfaForm` and `xml`, have their own data schema and are represented by nodes that have children. The children nodes represent data that is organized according to the variable's schema.

## Access to process data

You use process data for making decisions during process execution, for providing input data to service operations, and for saving service operation output data. Nodes in the data model can be retrieved using XPath expressions. The data model of a process can be viewed in XPath Builder, which is accessible from any view or dialog box that allows you to express a property value as an XPath expression.

RELATED LINKS:

Process design guidelines

Process execution

Creating variables

Process input and output data

Process data exposed to users

Variable types reference

# 5.5. Process design guidelines

This section provides the following general guidelines for designing processes:

## Order of implementation

Operational efficiencies can be gained when creating a process by implementing general details first, and addressing more specific details later. The following suggested workflow is effective for minimizing the effort required to support changes that are made to process diagrams when initially mapping the business process on the process diagram:

1) Create the process diagram using abstract modelling elements to represent the activities in the process.
2) After the activities of the process are represented and the routes are drawn to show the order of execution, replace the abstract elements with operations and event types.
3) Create variables and configure the properties of the operations as required.

   *This order of implementation is effective for the following reasons:*

   – Abstract modelling elements can be easier to work with when initially drawing the process diagram because the intended representation of activities is flexible.
   – The order in which operations are executed should be determined before they are configured because their order affects the flow of data, which informs the configuration of the operations. Changing the order before you configure the operations does not require reconfiguration of the operations.

     *Additionally, modelling the business process using abstract modelling elements requires a different skill set than implementing the model with service operations. These tasks can be performed by different team members that play different roles.*

## Process designs for reuse

To maximize resources at design time, as much as possible, you should develop processes so that they can be initiated by other processes.

When processes are activated, they are available as services that you can initiate from other processes. When a process's service is called from another process, the called process is a subprocess.

If more than one of your business processes include a common pattern of activities, you should implement the common activities as a separate process and use it as subprocesses. The end result of calling

the subprocess rather than including the common pattern of activities in each process is identical. However, when you use subprocesses, you only need to develop common patterns once.

If you are planning multiple processes at the same time, you can realize commonalities before you create the process diagrams and integrate the subprocess architecture into your plans immediately.

If you have already created a process and later realize that a new process that you are planning uses a common pattern of activities, you can implement the subprocess architecture retroactively. You can isolate the common activities from the existing process in a new process by copying. You then replace the common activities in the existing process with the subprocess's invoke operation.

### Reuse of variables

To make efficient use of RAM and space used in the AEM Forms database, when possible you should reuse variables to store process data. Each variable that you create uses RAM at run time, and, for long-lived processes, space in the database for persisting the data.

Reusing variables involves saving different data to a variable at multiple points the process. Each time new data is saved to the variable, the existing data is overwritten. Save new data to a variable only when the existing data is no longer required in the process.

For example, a process routes form data to a series of users in a process. Each time a person submits form data, the data is saved in the same xfaForm variable because the data that was submitted by the previous user is no longer required in the process.

To conserve memory and database storage, you can also keep the number of complex data types to the minimum required, and use simple data types where possible.

RELATED LINKS:

Process execution

Invoking subprocesses

Creating variables

Process life cycle

Variable types reference

Adding, defining, and deleting abstract activities

## 5.6. Process execution

This section provides information about the run-time behavior of processes:

### Process instances

Each time a process is initiated, the AEM Forms Server executes the process. Each execution of a process is called a *process instance*. Process instances have a unique identification and have their own process data associated with them.

For example, a process creates a credit card statement by retrieving information from a database, merging it with a form, and printing the statement to be mailed to a customer. The process is initiated monthly for each customer. Each time the process is initiated, a new process instance is created. The process data includes the information that is retrieved from the database, which is different for each customer.

### Process modifications

When you make changes to an activated process, the changes affect all process instances that are associated with it.

For example, you initiate several processes in the development environment to test the process you are creating. You then make a change to the properties of an operation on the process diagram. When you save the changes, they affect all process instances that have not yet progressed past that operation.

### New processes

## Process life cycle

When a process is initiated and the process instance is created, a branch instance is created for the main branch of the process. Subsequently, the operations that belong to the branch are executed in series, according to the design of the process.

When the execution of an operation is complete, the routes that begin at that operation are evaluated. The route evaluation determines which operation is executed next. When all of the operations in a branch are complete, the branch is complete.

If an error occurs when a route is being evaluated, the branch to which the route belongs is stalled. If an error occurs when an operation is executing, the operation is stalled. When a branch or operation is stalled, you can take measures to fix the cause of the error and use administration console to restart the branch or operation.

## Short-lived processes and long-lived processes

The Type property of processes can be set to long-lived or short-lived. In general, short-lived processes require less server resources at run time than long-lived processes. Configure processes to be short-lived when possible.

However, the value you specify affects how the process executes and the data that it saves at run time. It also determines the type of the main branch of the process and the type of branches that you can add to the process diagram by using gateway elements. You need to consider all of these factors when configuring the Type of the process.

**NOTE:** The value stored in the `id` node of the process data model is always -1 for short-lived processes. For long-lived processes, the value is an alphanumeric string. (See Processdata model.)

### Execution

The amount of time that it takes to execute a process is a factor that you need to consider when configuring the Type property:

- Configure your process to be short-lived if it takes very little time to execute (in the order of milliseconds). Short-lived processes are executed in a single transaction (see Transactions). Shorter execution times lower the risk of a system failure occurring during the transaction.
- Configure your process to be long-lived if one or more operations requires a long time to execute. Each operation in a long-lived process is executed in a separate transaction.
- You need to configure your process to be long-lived if the progression of the process depends on messaging from resources that act independently (outside) of process execution (asynchronous execution).

  *For example, processes that use the User service or event receivers should be configured as long-lived:*

- The User service's Assign Task operation is complete only after a user submits the associated task using Workspace.
- Event receivers and event start points wait for events to be thrown before the process continues.

### Client invocation

The Type property of the process affects how client software interacts with the process instance:

- When client software invokes short-lived process, the client software waits until the process instance is complete before continuing with its execution. When the process instance is complete, it returns the process output data to the client immediately.
- When client software invokes long-lived processes, the client software continues its execution immediately as the process instance is executing. The client software needs to retrieve the process output data after the process is complete.

### Data persistence

The Type property of the process determines the process data that is saved in the database. If your process passes data between operations, the data needs to be stored in the database. Also, your organization's policies may dictate that it is important to keep a record of everything that occurred during the process instance. In other cases, you may decide that saving storage space is more important than keeping detailed records:

- Configure the process to be long-lived if you need to pass data between operations or if you want to keep a complete record of the process instance.
- Configure the process to be short-lived if no data is passed between operations or you do not want to keep a record of the process instance.

| process type | Information stored |
|---|---|
| Long-lived | All process information is stored:<br>• Process instance data<br>*Branch instance data*<br>*Operation instance data* |
| Short-lived | No process information is stored. |

**NOTE:** Because short-lived processes do not store information in the database, fewer resources are required at run time, causing them to execute more efficiently.

**Branch types**

Long-lived and short-lived processes support different branch types:

- Short-lived processes support only transactional branches. The main branch in short-lived processes is a transactional branch.
- Long-lived process support asynchronous, synchronous, and transactional branches.

  **NOTE:** You cannot add gateway elements to short-lived processes. If you want to execute two or more branches in parallel, you need to use a long-lived process.

## Transaction Propagation

For short-lived processes, transaction propagation specifies whether the process executes in its own transaction or whether it executes in the transaction of the client software that initiated the process. The following values can be specified for the transaction propagation of a process:

**Mandatory:**

The process instance executes within the client's transaction. If the client is not associated with a transaction, an exception occurs.

**Never:**

An exception occurs if the client is running within a transaction. If the client is not associated with a transaction, a transaction is not created for running the process.

**Not Supported:**

The client’s transaction is suspended while the process instance executes. When the process instance is complete, the client’s transaction resumes. If the client is not associated with a transaction, a new transaction is not created.

**Required:**

The process instance executes within the client's transaction. If the client is not associated with a transaction, a new transaction is created for running the process.

**Requires new:**

The client's transaction is suspended. The process instance is executed in a new transaction. When the process instance is complete, the client's transaction resumes. If the client is not associated with a transaction, a new transaction is created for running the process.

**Supports:**

The process instance executes within the client's transaction. If the client is not associated with a transaction, a new transaction is not created for running the process.

The Transaction Timeout property of a process determines the amount of time that transactions have to complete execution before timing out.

RELATED LINKS:

Operation and branch compatibility

Creating processes using the New Process wizard

Transactions

Testing and troubleshooting process versions

Adding and deleting operations

Branches

Working with operations

Drawing routes to link operations

# 5.7. Process diagram modeling examples

This section describes how to implement common process constructs that occur in business processes. Each example in this section includes a description of the construct as well as a description of how to implement it in a process diagram.

The example process diagrams use the activity element as steps in the process because they can be replaced by any activity that an actual implementation requires. The patterns and examples that are described are not exhaustive. Many other constructs can be implemented.

## Sequential routing

In this example, an activity in a process occurs after the completion of another activity.

In the following illustration, activity1 and activity0 are routed sequentially, and activity1 executes after activity0.

RELATED LINKS:

Adding and deleting operations

Adding and deleting routes

## Parallel branches

In this example, a single branch diverges into multiple branches that execute in parallel. You implement this example using a gateway element that includes the branches that execute in parallel.

In the following illustration, branch0 and branch1 execute in parallel.

RELATED LINKS:

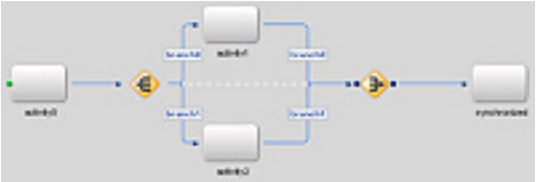
Synchronization

Adding branches using gateways

## Synchronization

In this example, branches that execute in parallel converge into a single branch. All branches execute to completion before the process continues after the convergence occurs. This example is implemented using a gateway element with the Control Type property set to AND-WAIT.

In the following illustration, the activity named synchronized does not execute until both branches in the gateway are complete.



RELATED LINKS:

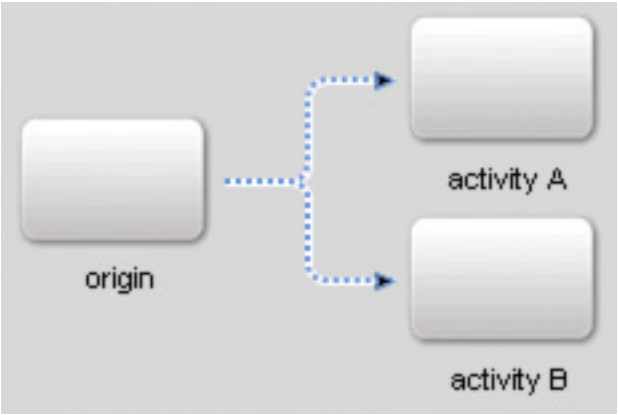
Adding branches using gateways

## Conditional routing

In this example, one of multiple possible routes follows an activity, requiring a decision to be made. This example includes the following features:

- Multiple routes originate at the same activity and terminate at different activities.
- Conditions are added to the routes that are evaluated to determine route validity.
- The order in which the routes are evaluated are configured on the activity where the routes originate.

  *In the following illustration, routing conditions are used to determine which route is followed and which activity is executed next. Only one of activity A and activity B executes, depending on the evaluation of the routing conditions.*




RELATED LINKS:

Simple merge

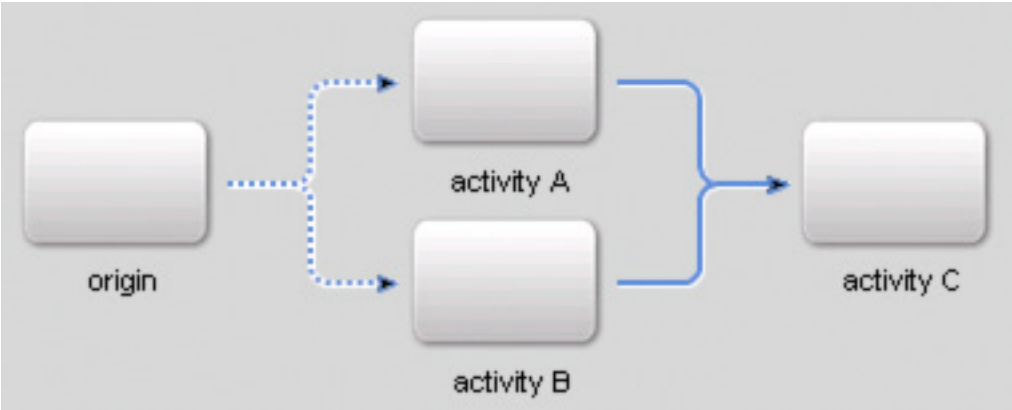
Adding and deleting routes

Making decisions using routes

## Simple merge

In this example, multiple routes converge at a single activity, but only one of the routes was followed. This construct occurs after the conditional routing construct. To implement a simple merge, multiple routes that originate at different activities terminate at a single activity.

In the following illustration, activity C executes after either activity A or activity B is complete.




RELATED LINKS:

Conditional routing

Adding and deleting routes

Making decisions using routes

## Multi-choice

In this example, more than one of several possible branches in a process are followed. This example can be implemented using either a gateway element or event receivers that are used as process start points.

### Gateway implementation

This implementation includes a process variable that contains information about which branch to execute and a gateway that contains the branches that can be executed. Each branch has the following characteristics:

- The first operation in each branch is an execute operation that the DecisionPoint service provides.
- The route that originates at the execute operation leads to the activities that are to execute in the branch.
- A condition is attached to the route to determine whether the activities should be executed. If the route is valid, the activities are executed. If the route is not valid, the branch completes and the activities are not executed.

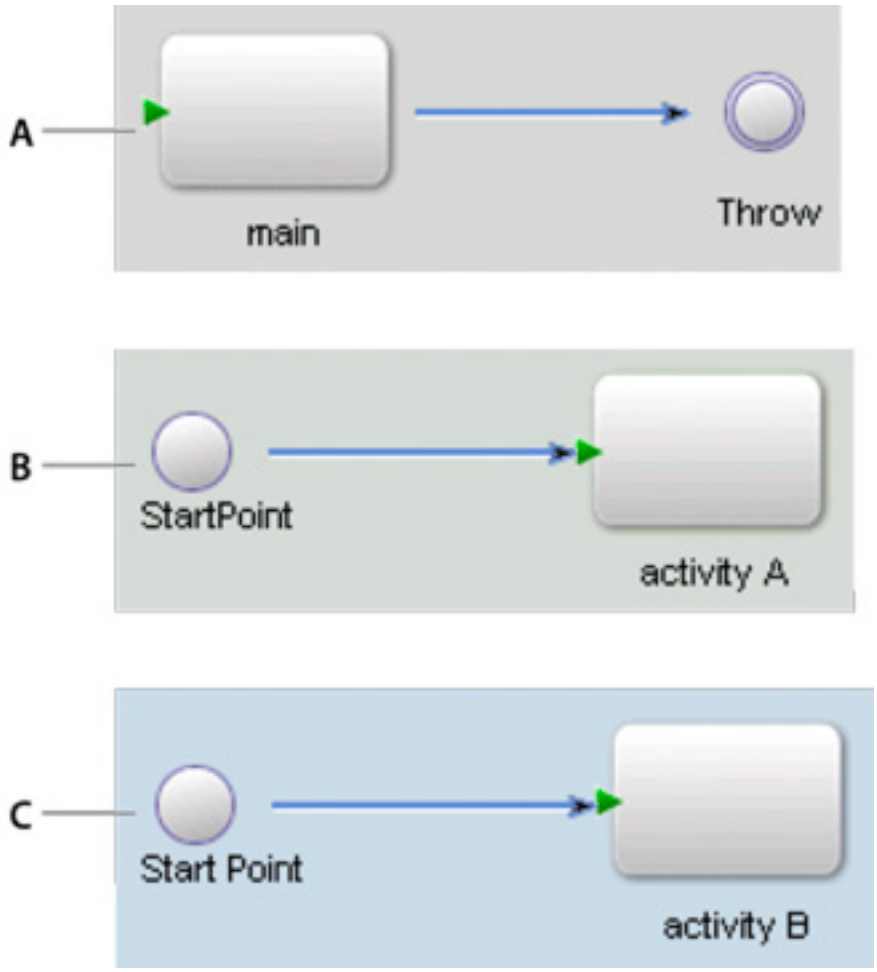
  *In the following illustration, activity A, activity B, or both activities are executed.*

### Event implementation

This implementation includes one main process and several subprocesses. The subprocesses are the possible branches that are followed. An asynchronous event type is used to determine which subprocesses are executed at run time:

- The main process throws the event. Event data is included in the event information.
- The subprocesses each receive the event as a start point. The filters on the start point determine whether the subprocess is executed.

  *In the following example, the main subprocess throws an asynchronous event, and the subprocesses use the event as a starting point. The filters on the start points of the subprocesses determine whether the subprocess is invoked, and therefore whether activity A, activity B, or both activities are executed.*




**A.**

Main process

**B.**

Subprocess containing activity A

**C.**

Subprocess containing activity B

RELATED LINKS:

Adding branches using gateways

Controlling process flow using events

Making decisions using routes

About Processes

## Synchronizing merge

This example is a synchronization that follows the construct described in the multi-choice example. The branches that execute during a multi-choice construct converge at a single activity, and the activity is executed only when all of the executed branches are complete.

The implementation of this example depends on whether you used gateway elements or events to implement the multi-choice construct.

### Implementation for a gateway multi-choice

The implementation of the synchronizing merge construct for a gateway multi-choice construct includes the following characteristics:

- The activity to execute after each branch executes follows the gateway.
- The value of the Control Type property of the possible gateway is set to AND-WAIT.

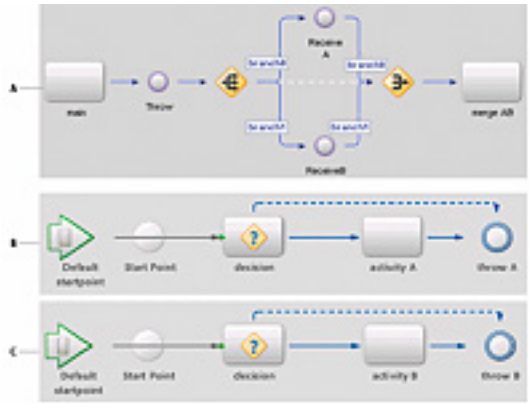
  *The following illustration shows the activity that follows the gateway and is executed only after all of the branches in the gateway are complete.*

### Implementation for an event multi-choice

To implement a synchronized merge for a multi-choice construct that is implemented using events, each subprocess needs to send a message back to the main process to indicate that they are either complete or that the subprocess was not executed:

- The start point of each subprocess does not filter the event data. Instead, an execute operation of the DecisionPoint service determines whether the activities in the subprocess are executed.
- Each subprocess throws an asynchronous event as the final activity of the subprocess.
- The event throw carries the name of the subprocess that it belongs to as part of the event data.
- The main process contains a gateway with a branch for each subprocess, and each branch includes a receiver for the event that each subprocess throws. The Control Type property of the gateway is set to AND-WAIT.

*The following illustration shows the main process that includes the mergeAB activity that executes after each subprocess is complete. The illustration also shows the subprocesses that throw the event to signal that the activity in the subprocess is either complete or did not execute.*



**A.**

Main process

**B.**

Subprocess containing activity A

**C.**

Subprocess containing activity B

RELATED LINKS:

Multi-choice

Synchronization

Adding branches using gateways

Controlling process flow using events

Making decisions using routes

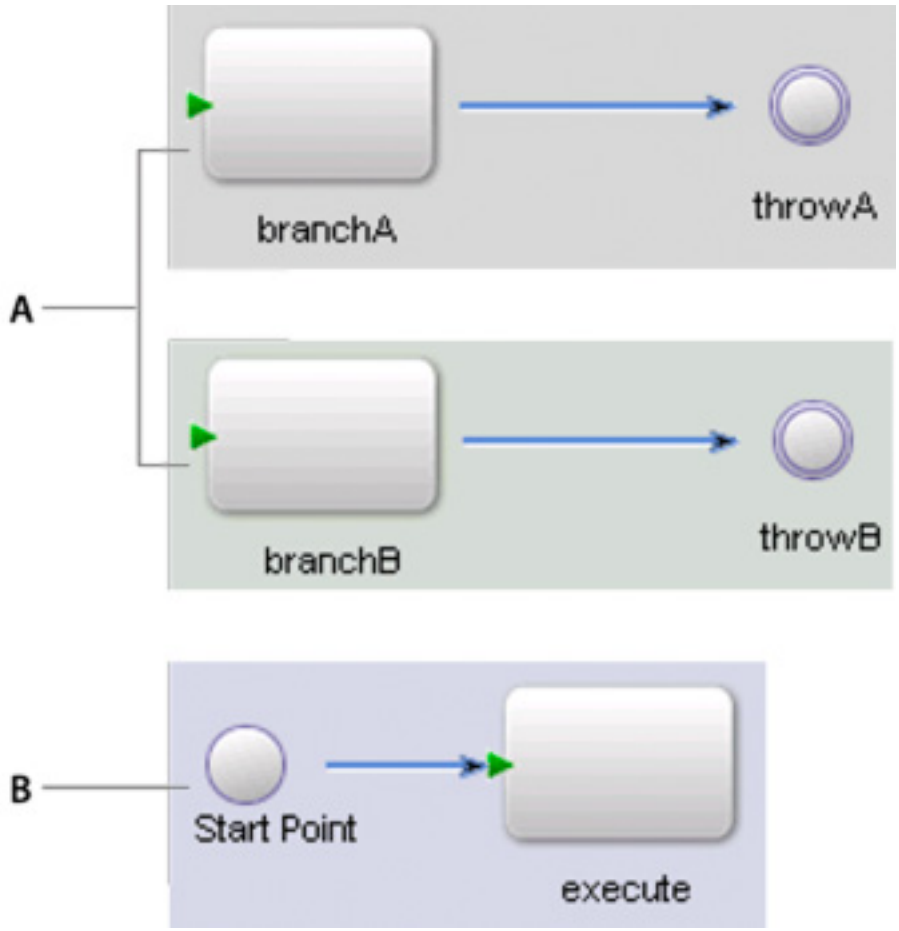
## Multi-merge

In this example, an activity is executed each time a branch that executed in a multi-choice construct is complete. This example is implemented using events and subprocesses:

- Each branch that executes in the multi-choice construct ends with the throw of an asynchronous event type. If the multi-choice construct used gateways, each branch in the gateways throw the event. If the multi-choice construct used events, the final step in each subprocess is a throw of the event.
- A subprocess that includes the activity to execute receives the event as a start point.

  *The following illustration shows two branches from a multi-choice construct that throws the asynchronous event type and the subprocess that uses the event type as a start point. The branches are represented as an activity element. The branches are inside either a gateway or a subprocess, de-*

*pending on the type of implementation used for the multi-choice construct that precedes the multi-merge construct.*




**A.**

Branches in the multi-choice construct

**B.**

Subprocess receives event throws

RELATED LINKS:

Multi-choice

Process designs for reuse

Controlling process flow using events

## Loops

In this example, a segment of a process diagram is repeated until a control condition is met. To implement this example, two routes originate at the final activity of the repeated segment:

- One route terminates at the first activity of the repeated segment.
- The other route terminates at the next activity after the loop.

- A condition on one of the routes is evaluated to determine whether the segment is repeated.

  *The following illustration shows a loop that contains two activities that are routed sequentially.*

**Loop counters**

In some situations, the loop needs to be executed a certain number of times. You can use a counter to keep track of the number of times the loop needs to be executed:

- The total number of loops to execute can be provided at run time or at design time.
- The number of loop iterations is stored in a variable (of type `int`).
- For each loop, the value of the variable is decremented by using the SetValue service.
- The value of the variable is used in the control condition to see whether the loop should be executed again.

RELATED LINKS:

Sequential routing

Process variables

Making decisions using routes

## Process completion

In this example, no valid route is found; therefore, the process is complete. This example is implemented in two possible ways:

- No routes originate at the final activity in the process. When the final activity completes execution, the process is complete.
- All of the routes that originate at an activity have a condition attached to them. The process completes when the conditions are evaluated and none of the routes are found to be valid.

No routes originate at final activity

All routes are conditional and none are valid

RELATED LINKS:

Adding and deleting routes

Making decisions using routes

## Multiple independent instances

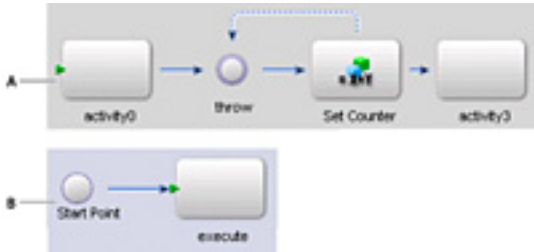
In this example, a given number of instances of an activity are executed, and the instances execute independently. This example is implemented using a loop with a counter, and either a gateway element or an event throw that is in the loop.

### Gateway implementation

The gateway contains a single branch that includes the activity or activities that you want to execute independently. The value of the Control Type property of the gateway is set to NO-WAIT so that the next iteration of the loop is executed without waiting for the gateway branch to complete.

### Event implementation

The loop includes a throw of an asynchronous event type. A subprocess uses the receiver of the event as a start point. Each invocation of the subprocess executes independently.




**A.**

Event throw is inside loop.

**B.**

Subprocess receives event as start point

RELATED LINKS:

Loops

Adding branches using gateways

Controlling process flow using events

## Multiple instances synchronization

In this example, multiple independent branches, which are executed in the multiple independent instances construct, converge at a single activity in the process. The implementation of this example involves using a loop and an asynchronous event type:

- Each branch instance, executed in the multiple independent instances construct, ends with the throw of an asynchronous event type.
- Following the loop that the multiple independent instances construct contains is another loop that includes a receiver for the event type. The receiver executes each time an instance of the independent activity completes.
- A counter keeps track of the number of times the receiver is executed.

  *The implementation of this example depends on whether you used gateway elements or events to implement the multiple independent instances construct.*

### Gateway implementation

The following illustration includes the following elements:

- An event throw at the end of the branch in a gateway (the multiple independent instances construct)
- An event receiver inside a loop that is used for tracking the number of branches that are complete

**A.**

Branch in multiple independent instances construct throws event

**B.**

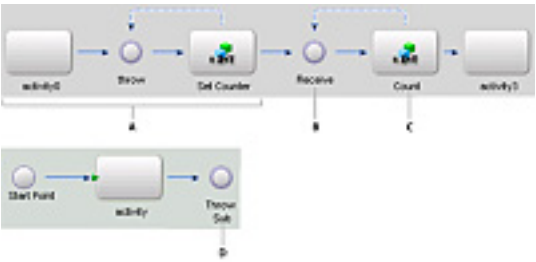
Multiple instances synchronization construct receives event

**C.**

Counts number of received events

### Event implementation

The following illustration shows the event throw at the end of the subprocess that is executed multiple times in the multiple instances construct. The main process receives the event inside a loop for tracking the number of subprocesses that are complete.




**A.**

In the main process, multiple independent instances construct

**B.**

Multiple instances synchronization construct receives event

**C.**

Counts the number of received events

**D.**

In the subprocess, throws event upon completion

RELATED LINKS:

Multiple independent instances

Loops

Adding branches using gateways

Controlling process flow using events

# 6. Process Quick Starts

Process Quick Starts provide information to help you implement frequently encountered business process scenarios in your process versions. Some Quick Starts focus on how to achieve results by using service operations from multiple services that are part of AEM Forms. Other Quick Starts focus on common tasks that are achieved by using other Workbench tools, which are useful for any of the available service operations that are part of AEM Forms.

## 6.1. Recommended skills

Before you use the process Quick Starts, you should already be familiar with basic process development skills:

- Adding operations and drawing routes
- Adding routing rules
- Creating variables
- Creating expressions
- Adding files to the Resources view

To learn these skills, you can work through the *Creating Your First AEM Forms Application* tutorial.

RELATED LINKS:

Gettingstarted with process design

AboutProcesses

## 6.2. Assigning tasks based on roles

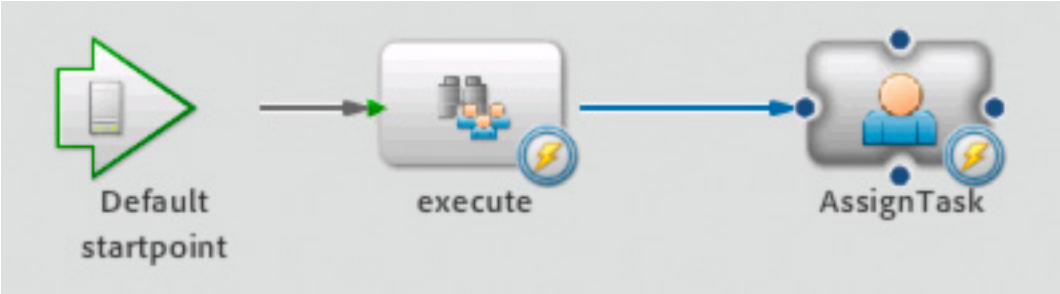
When using the User service to create human-centric processes, it is often advantageous to assign tasks to individuals based on their role in the process. This strategy is more flexible than assigning tasks to a specific person because, typically, different people are involved in different instances of the process.

This Quick Start discusses the use of the LDAP service, which is part of Foundation. It also discusses the User service, which you can use if you have forms workflow.

Typically, roles in processes correspond with roles in a company. You can leverage information in your LDAP server to assign tasks to users if the user records are organized according to the company's organizational structure or if the records themselves contain organizational information.

For example, a company's internal purchase-request process requires that a person's manager approve the purchase request before the purchase is made. The role of manager is constant; however, the person who requests the purchase determines who the manager is. The user records in the corporate LDAP server include the identification of the user's manager. The LDAP server can be queried to find out who that person's manager is who submitted the purchase request.

The process diagram that implements this business process can use the LDAP Query operation of the LDAP service to query the LDAP server. After the identification of the manager is retrieved, the task for approving the purchase request can be assigned to that manager. The following illustration shows the part of the process diagram that performs the LDAP query and the task assignment.




The following illustration describes the structure of the information in the LDAP server that the previously described example process uses.

```
o=company_name
 ├ ou=people, o=company_name
 │  └uid=username, ou=people, o=company_name
 │
 └ ou=groups, o=company_name
```

The records with the distinct name (DN) in the format uid=*username*, ou=*people*, o=*company_name* contain information about individual company employees. The record includes the following attributes:

**uid:**

The unique identification of the user

**manager:**

The DN of the employee's manager

## Prerequisites

For assigning tasks based on roles, you are typically retrieving the identification of a user based on some other information (for example, job title, business unit to which the user belongs, or both.

Several pieces of information need to be available at design time or at run time to use the LDAP service to perform queries:

- An understanding of how information is organized in the LDAP server. To perform the query, you need to specify a location in the LDAP schema to search.
- Specific information and its relation to the sought-after information in the LDAP repository. This information is used to create a search filter that identifies the data to retrieve.
- A `string` or `list` value that stores the retrieved information, depending on whether you are retrieving information for one or many LDAP records.

In our example process, the user identification of the person who requests the purchase is used in the query to find their manager.

## Configuration

To configure an LDAP Query operation, you need to identify the area of the LDAP schema that you are searching, the type of LDAP object that you are searching, and the search filter to use.

The Query tab of the LDAP Query Options Editor is configured to search for the record that represents the user who submits the purchase request:

**Base DN:**

`o=company_name` (the topmost level in the directory.)

**Search Context:**

`ou=people, o=company_name` (The level in the directory that contains the records that are being searched.)

**Search Filter:**

`(uid={$/process_data/@creator_id$})` (The example process is invoked when the person submits the purchase request, and the identification of the user is stored at the location `/process_data/creator_id`.)

**Search Scope: ONE LEVEL (The records below the value for Search Context.)**

The Output tab of the LDAP Query Options Editor is used to specify that the value of the `manager` attribute that the returned record contains is stored in a `string` variable named `stringVar`:

**Value:**

`manager`

**Location:**

`stringVar`

When the query is executed at run time, the `stringVar` variable contains the DN of the user's manager, which is in the format `uid=`*user id*`, ou=people, o=company_name`. The task that the Assign Task operation generates needs to be assigned to the user that this DN represents.

The Select Initial User property of the Assign Task operation is assigned the *user id* part of the manager's DN (the user identification). An XPath expression that resolves to the `stringVar` variable is used. The `substring-after` and `substring-before` functions are used to isolate the *user id* portion:

```
substring-after(substring-before(/process_data/@stringVar, ","),"uid=")
```

### Other considerations

This Quick Start does not show the initiation step of the process. Also, this Quick Start focused on the LDAP Query operation. The LDAP service also provides the LDAP Query to XML operation that you can use to store query results in XML format.

RELATED LINKS:

Gettingstarted with process design

AboutProcesses

LDAP

SetValue

CreatingXPath expressions

## 6.3. Using form data with multiple forms

When using the User service to involve people in processes, you may be required to use more than one form in a process and populate them with the same form data. If all forms use the same layout for the form fields, they can be used interchangeably with the same form data. However, if two form designs use different layouts, form data from one form cannot be automatically merged with the other form. In this case, you need to manipulate the form data so that it properly fills the form.

This Quick Start will be most useful to you if you are already familiar with using forms in processes.

For example, a person fills an order form to purchase parts from a retailer's online automobile parts catalog and submits the form to initiate the purchase process. The process determines that some of the ordered parts are not in stock and must be ordered from the manufacturer. The process automatically fills a different form to order the parts from the manufacturer and fills this form with data from the customer's original purchase request.

You can use these strategies to manipulate form data to use it with a different form layout:

- Make changes to the form data (which is stored in an xfaForm variable) so that the data structure matches that of the form. Use this strategy if the forms use only slightly different field layouts so that adding nodes to the data schema makes them compatible.
- Copy data items from the form data to an xfaForm variable that uses a data structure that is compatible with the form's data schema. Use this strategy when the forms use very different field layouts or if you only need to use a subset of the form data.

**TIP:** To implement either strategy, use the Set Value service to manipulate the data stored in form variable.

**TIP:** T*o see the form data schemas in XPath Expression Builder, embed the schema in the form.*

### Prerequisites

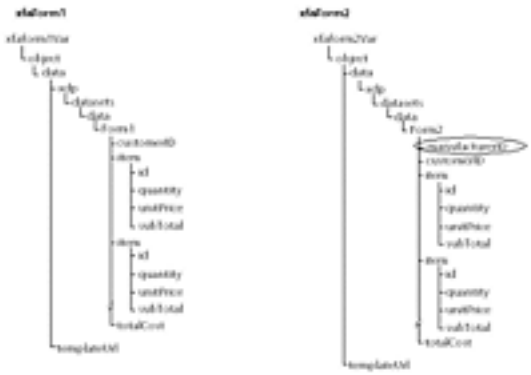
To use the same data for two different forms, you need to define two xfaForm variables that hold the form data you are manipulating:

- One form variable holds the original form data.
- One form variable holds the form data that is based on the original form data but has been manipulated.

## Configuration when forms are similar

If two forms use field layouts that are very similar, you may be able to add nodes to the data from one form so that the data can be successfully merged with the second form. You can use XPath expressions to add nodes to the data model of xfaForm variables.

The following illustration shows the data models of two different xfaForm variables (xfa`form1Var` and xfa`form2Var`) for populating two different forms (Form1 and Form2).



The difference between the two schemas is that the schema for the `xfaform2Var` variable includes the `manufacturerID` node below the root node of the form schema. The data in the `xfaform1Var` variable is made compatible with Form2 by adding a `manufacturerID` node to its schema.

The following expression adds the `manufacturerID` node to the data schema of `xfaform1Var`:

```
/process_data/xfaform1Var/object/data/xdp/datasets/data/Form1/manufacturerID
```

This expression is used as the value of the location attribute for a Mapping property of a Set Value service's execute operation to create the `manufacturerID` node.

**NOTE:** To preserve the original form data, you must copy the data to another variable before you modify the data schema.

## Configuration when forms are not similar

You can copy data from the nodes of one xfaForm variable data model to the nodes of another xfaForm variable schema. If two forms use field layouts that are very different, you can copy data items from one form variable to the other form variable so that the data is reused in the second form.

The following illustration shows the data schemas that are required in two different form variables (`firstVar` and `secondVar`) for populating two different forms (firstform and secondform).

The form called *secondform* contains fewer fields than firstform so that the data captured using firstform cannot be automatically merged with secondform. However, the data from some firstform fields are appropriate for fields on secondform. You use the Set Value service to copy items of data from the `firstVar` variable to the `secondVar` variable.

For example, you would use the Set Value service to set the value of the `/process_data/secondVar/object/data/xdp/datasets/data/secondform/item[1]/id` node in the `secondVar` variable to the value of the `/process_data/firstVar/object/data/xdp/datasets/data/firstform/item[1]/id` node of the `firstformVar` variable.

### Other considerations

The examples in this Quick Start use simple XPath expressions. If required, you can manipulate data with more complex XPath expressions and use XPath functions.

RELATED LINKS:

Gettingstarted with process design

Processdata

CreatingXPath expressions

Designinghuman-centric processes

xfaForm

## 6.4. Assembling multiple documents

You can create new PDF documents from other existing PDF documents by using the invokeDDX operation of the Assembler service. Typically, multiple existing documents are selected from a library and assembled as required at run time.

For more information about the Assembler service, see Services ReferenceforAEM Forms.

The Assembler service requires the following items to assemble PDF documents:

- A Document Description XML (DDX) document that provides instructions for assembling PDF documents

- The PDF documents to be assembled
- A location to store the resulting PDF document

For example, a business process assembles several existing PDF documents into a single PDF document, and adds a cover page to the document. The following illustration shows a portion of the process diagram that automates the process.

In this example, the DDX file that is used to assemble the documents is stored in the repository. At run time, the DDX file is retrieved and used as input data for the Assembler service. The documents to assemble are provided to the process when the process is invoked. Press F1 for more information.

## Prerequisites

To use the invokeDDX operation, the following items need to be available to the process at run time:

- The DDX document, in `document` data format.
- The PDF documents to assemble, in `document` format.
- The key words that are associated with the documents to assemble and the resulting PDF document. The keywords are defined in the DDX document.

For the example process described above, the documents to assemble are provided as input values when the process is invoked, and the DDX document is retrieved from the repository by using the Repository service. The key words that are associated with the documents by the DDX document are known at design time.

## Configuration

The invokeDDX operation of the Assembler service requires the following data items as input:

- A `document` value that represents the DDX document, provided as the value of the DDX property of the operation.
- A `map` value that holds the documents to be assembled, provided as the Input Document Map property of the operation. The keys for the `map` value are the key words that the DDX document defines. Each value in the `map` are either a `document` value that represents a single PDF document or a `list` of `document` values for representing multiple PDF documents.

Typically, the PDF documents to assemble are available at run time as `document` values. In this case, a `map` value needs to be created and populated with the document values so that it can be used as the value of the Input Document Map property of the invokeDDX operation.

The DDX document and the number of PDF documents to assemble determine whether the `map` value should contain `document` values or `list` values that hold `document` data items. For example, the following DDX document is used for the example process described above.

```
<?xml version="1.0" encoding="UTF-8"?>
<DDX xmlns="http://ns.adobe.com/DDX/1.0/">
<PDF result="GeneratedDocument.pdf">
<PDF source="cover"/>
<PackageFiles>
<PDF source="attachments"/>
</PackageFiles>
</PDF>
</DDX>
```

The DDX document provides the following information about the documents to assemble:

- The `<PDF source="cover"/>` element defines the `cover` key word, which is associated with the PDF document to use as the cover page.
- The `<PDF source="attachments"/>` element defines the `attachments` key word, which is associated with several PDF documents to assemble.

The `map` value must use `cover` and `attachments` as keys. The values for the keys must be `list` data items that hold `document` values:

- Multiple PDF documents are associated with the `attachments` key word; therefore, the value for the `attachments` key needs to be a `list`data item that holds the document values.
- Data items in a `map` must be of the same type; therefore, although only one document is used as a cover page, the value for the `cover` key needs to be a `list` data item.

### Retrieving the DDX File

In the example process diagram illustrated previously, the Read DDX File From Repository operation is a Read Resource Content operation that the Repository service provides:

- The Resource URI property of the Input property group is used to specify the location of the DDX document in the repository.
- The Result property is used to save the retrieved DDX document in a `document` variable.

### Assembling the document

In the example process diagram illustrated previously, the invokeDDX operation needs to be configured so that it uses the DDX document that was retrieved from the repository and the PDF documents that are provided to the process as input data.

- The value of the DDX property is specified as the value of the `document` variable that holds the DDX document.
- The `inputs` property of the Input property group is used to identify the `map` variable that stores the documents to assemble:
  - The cover key is associated with a list variable that contains one document to use as the cover page of the assembled document.

  - The attachments key is associated with a list variable that contains all of the other documents that are to be assembled.

- The value of the Result property of the Output property group is used to identify the `map` variable that stores the results of the invokeDDX operation. In this case, because only one document is assembled, the map includes one key/value pair. The key is `result`, as it was defined by the `<PDF result="GeneratedDocument.pdf">` element in the DDX document. The value is a `document` value that represents the assembled PDF document.

### Other considerations

This Quick Start uses a simple use case for describing how to use the Assembler service. However, the Assembler service can be used to manipulate PDF documents in many ways, such as extracting and inserting pages, disassembling documents based on bookmarks, and adding watermarks. Different combinations of document manipulations can also be performed.

RELATED LINKS:

Gettingstarted with process design

Assembler

document

map

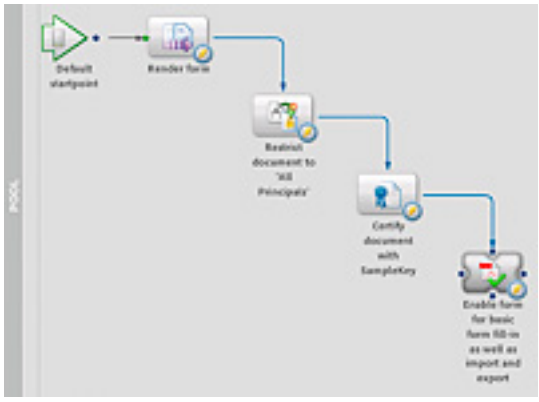
## 6.5. Certifying policy-protected documents

Using certified forms that are secured using the Rights Management service is useful when you want to control who can use the form and also allow form users to verify the form's authenticity.

This Quick Start will be most useful to you if you are already familiar with the Rights Management service. For more information, see Services ReferenceforAEM Forms.

For example, a company's purchase order process enables users to fill a PDF form and submit it online. The form is provided in PDF and is protected with an Rights Management 11 policy so that only registered users can use it. The form is also certified so that users can verify the form's authenticity.

The following illustration shows the process diagram that is used to prepare the form before it is provided to users.




The above process diagram includes the following service operations:

1) The renderPDFForm operation (Forms service) creates a PDF form from an XDP file. The name of this operation on the process diagram is *Render Form.*
2) The Protect Document operation (Rights Management service) applies a policy to the form to implement usage rights. The name of this operation on the process diagram is *Restrict Document To 'All Principals'.*
3) The Certify PDF operation (Signature service) certifies the document. The certification signature assures the recipient of the form's authenticity and integrity. The name of this operation on the process diagram is *Certify Document With Sample Key.*

**NOTE:** The order in which these operations are executed is very important. Policy protection of a PDF document must occur before certification.

## Prerequisites

To policy-protect and certify a rendered PDF form, the following conditions must be met when the process executes at run time:

- The form and, optionally, the initial data to merge with the form is available to the process.
- The form includes a signature field so that it can be certified.
- The policy that is used to protect the form has been created, and the name of the policy is available to the process. You use administration console to create the policy.
- The credential that is used to certify the form and the credential's alias is available to the process. You use administration console to import credentials into AEM Forms.

In the example process described previously, the form (XDP file) is stored in the repository, and the form data is provided as input when the process is invoked. Upon invocation, the form data is saved in a `document` variable.

## Configuration

This section describes how to configure the service operations that the example process diagram includes. The operations in the example process diagram build on the results of previous operations. Therefore, the output data from one operation is used as the input data of the subsequent operation.

Specifically, a document variable is used to store the PDF form that is passed between operations. This variable is also configured as output data for the process.

The renderPDFForm operation requires the form and form data as input:

- An xfaForm variable is configured to reference the XDP file that is stored in the repository. This variable is used as the value of the Form To Render property of the operation.
- The document variable that stores the initial form data is used as the value of the Form Data property.

The PDF form that the renderPDFForm operation creates is stored in a document variable. This document variable is also used as input for the Apply Policy operation so that the PDF form is policy-protected:

- The document variable is used as the value of the Input PDF Document property of the operation.
- The name of the policy that was created using Administration Console is used as the value of the Policy Name property of the operation.
- The policy set to which the policy belongs is used as the value of the Policy Set Name property of the operation.

The resulting policy-protected PDF form is stored in the same document variable that was used as the operation's input. The document variable is then used as input for the Certify PDF operation:

- The document variable is used as the value of the Input PDF Document property of the operation.
- The alias of the credential that was imported to AEM Forms using administration console is used as the value of the Alias property of the operation.

The certified PDF form is stored in the same document variable that was used as the operation's input. Because the document variable is configured as output data for the process, when the process is complete, the policy-protected and certified PDF form is returned to the user that invoked the process.

## Other considerations

The example process described in this Quick Start reused the document variable for saving the results of each operation. This practice minimizes storage space used in the AEM Forms database. If your processes need to preserve the results of each operation, save the results in different variables.

This Quick Start did not describe how to create Rights Management policies or import credentials into AEM forms by using Administration Console. For information about how to perform these tasks, see Rights Managementadministrationhelp and Trust Store ManagementHelp.

RELATED LINKS:

Gettingstarted with process design

Forms

RightsManagement

SetValue

Signature

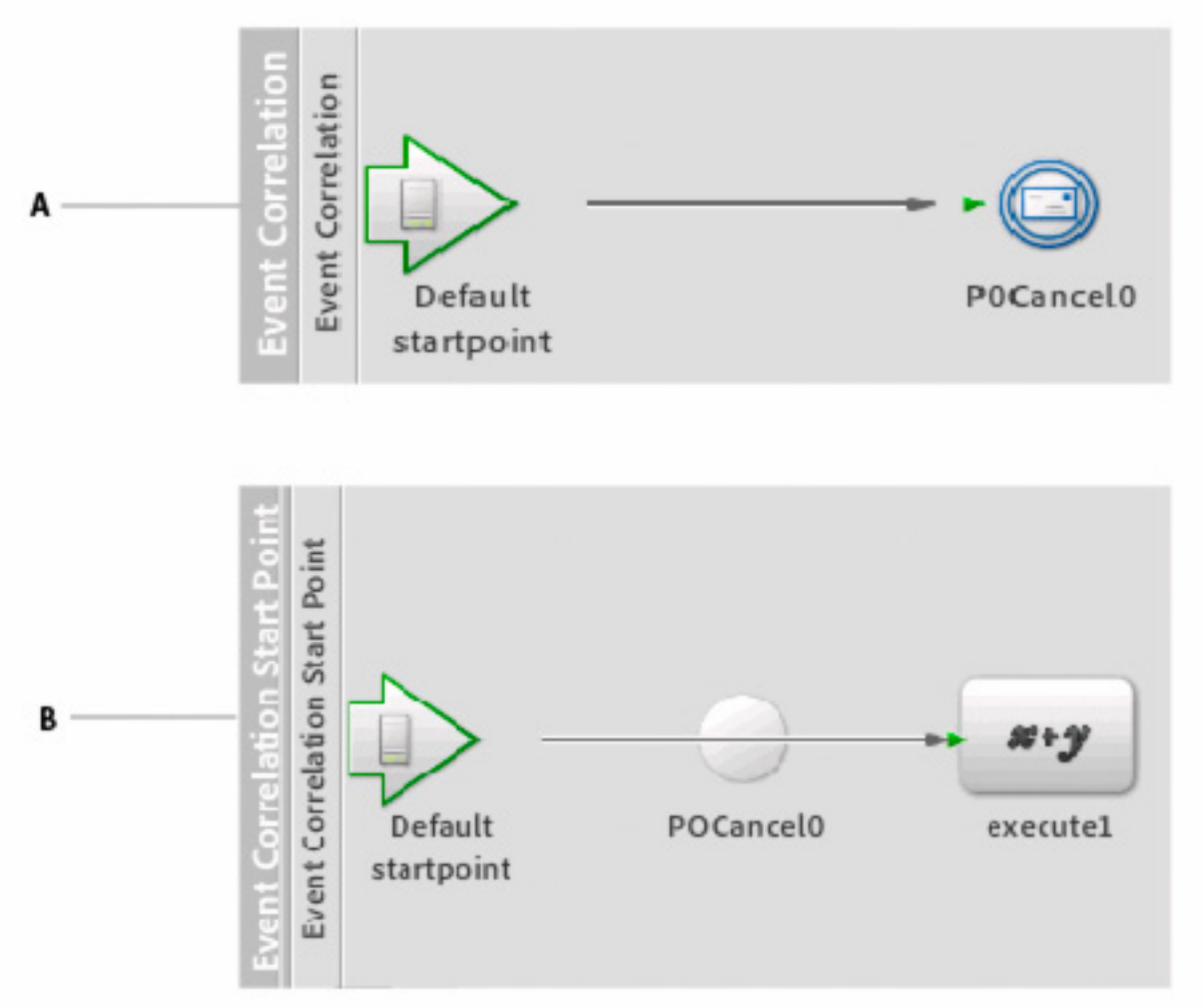
# 6.6. Throwing events to initiate processes

Events can be used for communicating between running processes. For example, you can use events to initiate or terminate a process, or to signal when processes can proceed when they depend on another process to complete a set of operations.

This Quick Start describes how asynchronous events can be thrown in a process for initiating another process.

The process that throws the asynchronous event includes the event throw of an event type. The type of event that should be used is determined by the context of the process. For example, when Workspace 9 users have attached a file to their task, a TaskAttachmentAdded event can be used.

The process that is initiated when the event is thrown includes the event start point of the event type. The following illustration shows a process that throws an event type, and another process that uses the event type as a start point.




**A.**

Process that includes the event throw.

**B.**

Process that includes the event start point.

## Prerequisites

To throw event types and use them as start points, the following conditions must be met at design time or when the process executes at run time:

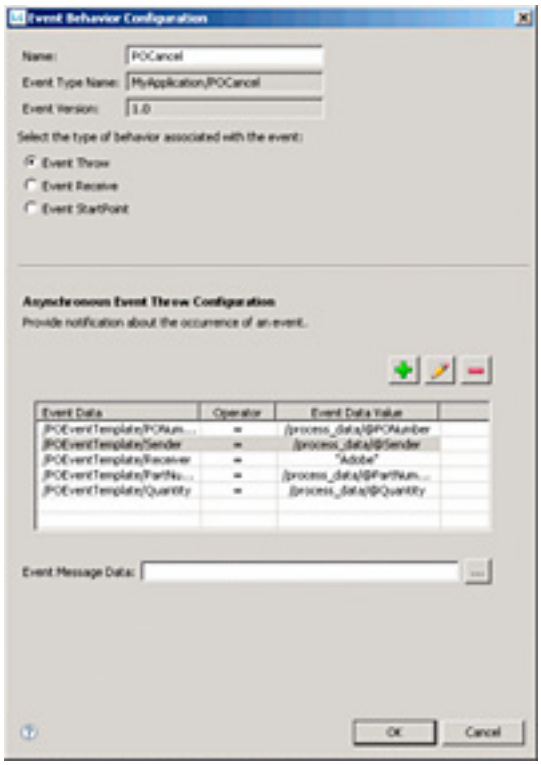
- The event type that you want to use needs to be available on the Events view.
- The information that you store in the event data set must be available to the process before the event is thrown. When you throw an event, you populate the event data set with information that event receivers and start points filter on.
- The variables that store event information have their Input property selected. When you use an event start point, you can store received event information in process variables.

The example processes described previously use an asynchronous event type named *POCancel*.

## Configuration

In the Event Correlation example process, the event data of the POCancel event throw is populated with literal values. For example, the `/POEventTemplate/Sender` item of the event data set is set to the literal value `Sender`.

Event data sent for the event throw.



In the Event Correlation Start Point example process, an event filter is configured for the POCancel start point so that the Event Correlation Start Point process is executed only if the event throw has a value of `"Akira Tanaka"` for the `/POEventTemplate/Sender` item.

Event filter for the event receive.

## Other considerations

When you use events, you first need to decide which event type to use. If an event type does not exist that meets the requirements of your process, you can create a custom event type.

Custom event types require that you create a data schema. You create a data schema using an XML Schema Definition (XSD) file. Optionally, you can include event message data by defining a separate XSD file, which is useful when you want to pass data from one process to another.

The example process diagram uses the event throw as the only step in the process. However, the event throw can occur between a series of operations in the process, according to the requirements of the process.

Also, the example event throw used literal values to populate the event data. However, a more typical implementation uses process data that is gathered at run time as event data. When run-time data is used, more meaningful filters can be configured on the event start point for deciding whether the process executes.

RELATED LINKS:

Controllingprocess flow using events

Throwingevents

Eventstart points

Addand configure timer event throws

## 6.7. Using Barcode Data in Processes

Using the barcoded forms service in processes, you can extract barcode data from scanned images (TIF or PDF files), convert the data to XML format, and use it in the processes.

This Quick Start is most useful if you are already familiar with the barcoded forms service. For more information about this service, see Services ReferenceforAEM Forms.

For example, a university's business process uses a barcoded form that students use to change either their name or their status for the school records. Two barcodes on the form capture the information that students enter on the form:

- One barcode captures information from a drop-down menu that the student uses to indicate whether they are changing their name or their status.
- The other barcode captures the details about their name or status change.

The barcoded forms service is used to read the barcode and process the information that it contains.

The following illustration shows the process diagram that is used to read the barcode and process the data. The data is processed differently depending on whether the student is changing their name or their status. The routes that follow the Extract Content operation have conditions that determine how to process the data accordingly.

The Decode Form operation reads the barcodes and saves the data as XML. The data includes metadata about the barcodes as well as the form data. The form data is in delimited-text format contained in `content` elements.

The following XML is an example of the output that the Decode Form operation produces. The data from each barcode is contained in separate `barcode` elements:

- The `barcode` element with the `id` attribute value of `1` represents the barcode that contains the details about the name or status change.
- The `barcode` element with the `id` attribute value of `2` represents the barcode that contains the value of the drop-down menu that indicates whether a name or status change has been submitted.

```
<?xml version="1.0" encoding="UTF-8"?>
<xb:scanned_image
xmlns:xb="http://decoder.barcodedforms.adobe.com/xmlbeans"
path="tiff" version="1.0">
<xb:decode>
<xb:date>2007-07-25T11:48:22.639-04:00</xb:date>
<xb:host_name>Win2K3-LC8</xb:host_name>
```

```
<xb:status type="success">
<xb:message/>
</xb:status>
</xb:decode>
<xb:barcode id="1">
<xb:header symbology="pdf417">
<xb:location page_no="1">
<xb:coordinates>
<xb:point x="0.119526625" y="0.60945123"/>
<xb:point x="0.44457594" y="0.60945123"/>
<xb:point x="0.44457594" y="0.78445125"/>
<xb:point x="0.119526625" y="0.78445125"/>
</xb:coordinates>
</xb:location>
</xb:header>
<xb:body>
<xb:content encoding="utf-8">t_SID t_FirstName t_MiddleName
t_LastName t_nFirstName t_nMiddleName
t_nLastName &#13;90210 Patti Y Penne
Patti P Prosciutto
</xb:content>
</xb:body>
</xb:barcode>
<xb:barcode id="2">
<xb:header symbology="pdf417">
<xb:location page_no="1">
<xb:coordinates>
<xb:point x="0.119526625" y="0.825"/>
<xb:point x="0.44457594" y="0.825"/>
<xb:point x="0.44457594" y="0.9167683"/>
<xb:point x="0.119526625" y="0.9167683"/>
</xb:coordinates>
</xb:location>
</xb:header>
<xb:body>
<xb:content encoding="utf-8">t_FormType t_FormVersion
&#13;ChangeName 20061128
</xb:content>
</xb:body>
</xb:barcode>
</xb:scanned_image>
```

Because the form data is delimited text, it is difficult to parse and extract information. Therefore, the Extract Content operation is used to parse and save the data in XML format. In XML format, XPath expressions can be used to access specific items of data.

The following XML is an example of the output that the Extract Content operation produces for the first barcode on the form (`id` = `1`).

```
<?xml version="1.0" encoding="UTF-8"?>
<xdp:xdp xmlns:xdp="http://ns.adobe.com/xdp/"
```

```
xmlns:xfa="http://www.xfa.org/schema/xfa-data/1.0/">
<xfa:datasets>
<xfa:data>
<form1>
<t_SID>90210</t_SID>
<t_FirstName>Patti</t_FirstName>
<t_MiddleName>Y</t_MiddleName>
<t_LastName>Penne</t_LastName>
<t_nFirstName>Patti</t_nFirstName>
<t_nMiddleName>P</t_nMiddleName>
<t_nLastName>Prosciutto</t_nLastName>
</form1>
</xfa:data>
</xfa:datasets>
</xdp:xdp>
```

The following XML is an example of the output that the Extract Content operation produces for the second barcode on the form (`id` = `2`).

```
<?xml version="1.0" encoding="UTF-8"?>
<xdp:xdp xmlns:xdp="http://ns.adobe.com/xdp/"
xmlns:xfa="http://www.xfa.org/schema/xfa-data/1.0/">
<xfa:datasets>
<xfa:data>
<form1>
<t_FormType>ChangeName</t_FormType>
<t_FormVersion>20061128 </t_FormVersion>
</form1>
</xfa:data>
</xfa:datasets>
</xdp:xdp>
```

The routes that follow the Extract Content operation include conditions that determine the next operation to execute. The conditions use the content of the `t_FormType` element from the second barcode to determine which route is valid. The content of this element represents the option that the student selected on the form for changing their name or their status. Because the data is in XML format, the content can be retrieved using XPath expressions. For example, the following XPath expression retrieves the content of the `t_FormType` element:

```
/process_data/contentList[2]/xdp:xdp/xfa:datasets/xfa:data/form1/t_FormT
ype
```

**TIP:** `contentList` is a `list`*variable that contains XML data. In this example, the variable stores the output of the Extract Content operation.*

## Prerequisites

To use the Decode operation, the barcode must be available to the process at run time. The barcode can be provided as an image file (TIF format) or the filled PDF file that contains the barcode can be provided (flattened PDF). A PDF document that was created from a scanned image of the barcode can also be used.

The barcoded forms service supports the following TIF formats:

- binary
- grayscale
- RGB color
- LZW compressed
- JPEG compressed
- CCITT Group 3 and Group 4 compressed

For the example process, a watched folder endpoint can be created for invoking the process. The file that contains the barcode is copied to the watched folder to create a process instance.

## Configuration

The Decode operation in the example process is an instance of the Decode operation that the barcoded forms service provides. The operation requires the following data items as input:

- A `document` value that represents a PDF document or a TIF image that contains one or more barcodes to decode.

  *When a watched folder is used as the process endpoint, the file that is provided is automatically converted to a `document` value.*
- The symbology used for the barcode.

  *The example process uses PDF417 barcodes.*

The results of the Decode operation are provided as XML data. An `xml` variable must be created and used to store the data.

The Extract Content operation in the example process is an instance of the Extract to XML operation that the barcoded forms service provides. The operation requires the following data items as input:

- An org.w3c.dom.Document value that represents the XML document that the Decode operation produces.

  *The xml variable that stores the output of the Decode operation in the example process can be used as this input value.*
- Information about the delimiters that are used in the barcode data.

  *In the example barcode data, the Tab character is used to separate field values and the carriage return character is used to separate lines of field names and field values.*

The results of the Extract to XML operation are provided as a `list` value that contains `xml` values. Each item in the `list` represents the data from each barcode that was processed.

## Other considerations

Form developers should select a simple format for delimiting barcode data to maximize the compatibility of the data with the Convert to XML operation. For example, the form design features of Workbench allows you to encode barcodes with each field name on a single line and the field value on the following line, using tabs to delimit the fields. This format enables process developers to use the default value of Tab for the field delimiter in the Extract to XML operation.

When you add the Decode operation to a process diagram, the operation catches an exception event by default. You can use the event catch to perform alternative activities if an error occurs when the operation executes.

## 6.8. Sending output to a printer

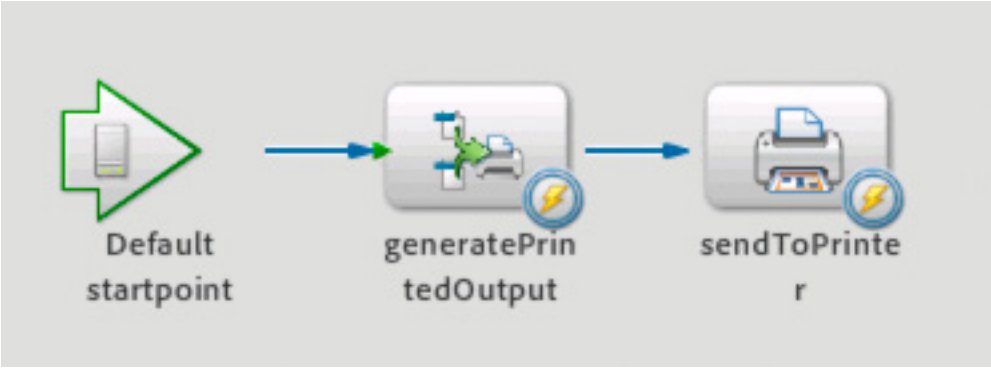
This Quick Start describes how to use the Output service to merge a form design with one or more records and send the output to a printer.

Using the Output service, the records can be merged with a form design to create filled forms that can be sent to a printer. The form design must have corresponding field names for each field to merge with the record. For example, when a purchase order process completes at an organization, a record that contains the contents of the purchase order is created and saved as an XML file. The contents of the XML file are merged with a form design. Each record that is merged with the form design creates a filled purchase order form. The filled purchase order is sent to a printer named *myprinter1* on a print server named *myprintserver1*. (The printing environment is configured by using Microsoft server networking.)

An application named *poPrintApp* implements this service for the organization. When a purchase order is completed, its contents are saved as a record in an XML file named poData.xml. The poData.xml file is sent at various time intervals to invoke the poProcessPrint service. These assets are in the poPrintApp application:

- A form design named PurchaseOrder.xdp that is used for merging with the XML data.
- A process named poPrintProcess that includes the following items:
  - An input `document` variable named *inputXML* that passes the XML data for purchases into the process.
  - An output `document` variable named *outputPS* that stores the merged PostScript® output.
  - A generatePrintedOutput operation (Output service), which merges a form design with records in an XML file. The resulting output is a `document` value that represents the output for printing.
- A sendToPrinter operation (Output service) that sends the `document` value to a printer. You can choose from various printer protocols, depending on how your printer settings are configured with the AEM Forms Server.

The process diagram for the poPrintPrint process looks like the following illustration:



The following XML file is generated from a purchase order request. The XML file can have one or more records in it. To determine where a record starts and begins, knowledge of which node or level the record starts at is required. In the following XML file, `<batch>` is the first node (or root node) and is therefore at level 1. The batch element contains three records. Each record is delimited by the `<PurchaseOrder>` node. Therefore, in the following example, each record starts at level 2.

```
<?xml version="1.0" encoding="UTF-8"?>
<batch>
<PurchaseOrder>
<lastName>Blue</lastName>
<firstName>Tony</firstName>
<OrderNo>555666777</OrderNo>
<Item>
<Description>Widget S</Description>
<Price>100</Price>
<Quantity>2</Quantity>
</Item>
<Item>
<Description>Widget B</Description>
<Price>5.00</Price>
<Quantity>4</Quantity>
</Item>
<PurchaseAmount>40.00</PurchaseAmount>
</PurchaseOrder>
<PurchaseOrder>
<lastName>White</lastName>
<firstName>Sam</firstName>
<OrderNo>555666222</OrderNo>
<Item>
<Description>Widget ABC</Description>
<Price>2200.00</Price>
<Quantity>10</Quantity>
</Item>
<PurchaseAmount>22000.00</PurchaseAmount>
</PurchaseOrder>
<PurchaseOrder>
<lastName>Green</lastName>
<firstName>Steve</firstName>
<OrderNo>55566688</OrderNo>
<Item>
<Description>Widget G</Description>
<Price>700000</Price>
<Quantity>1</Quantity>
</Item>
<PurchaseAmount>700000</PurchaseAmount>
</PurchaseOrder>
</batch>
```

## Configuration

For the generatePrintedOutput operation, the following properties are configured to generate PostScript output:

### Input

- **Form:** poPrintApp > poPrintApp/1.0 > purchaseOrder.xdp. Use the Select Asset dialog box to specify the literal value.
- **Input Data:** inputXML variable. The inputXML variable is an input `document` variable. The contents of the `document` value are from the poData.xml file.
- **Print Format:** Generic Postscript Level 3 because PostScript output is required. Depending on the print format, different default device profiles (XDC files) are used. Because Generic Postscript Level 3 is selected, the XDC URI property changes to ps_plain_level3.xdc. XDC files are printer description files in XML format that describe the capabilities of printers. You can modify XDC files. (See *Creating Device Profiles Using Workbench* .)

### Batch Options

- **Record Level:** 2, because each record begins at the second XML node in the XML file.

### Output

- **Printed Output:** The output generated by the operation. Save to the variable outputPS, which is `document` variable.

  *For the sendToPrinter operation, the following properties are configured to send to a printer (Microsoft network) that the AEM Forms Server can access:*

### Input

- **Input Document:** The `document` variable outputPS, which contains the PostScript output that was created from the generatePrintedOutput operation.
- **Printer Protocol:** SharedPrinter, because the printer is connected to the AEM Forms Server by using Microsoft server networking. The printer protocol to use depends on how printers are configured in the network.
- **Server URI:**\\myprintserver1, which is the name of the print server.
- **Printer Name:** \\printserver1\myprinter1, which is the name of the printer. The complete name of the printer name must include the printer server.

## Other considerations

This Quick Start does not show you how to create the XML that contains the collection of records or how to create a form design that can be used with XML data. For information about creating form designs, see Designer Help. For information about creating XML data and considerations for creating form designs for the Output service, see Designer Help.

The AEM Forms Server must be configured to access the printer, which requires you to set permissions on the application server. For more information about the configuration settings required to send output to print, see "Sending documents to printer" in the Output Service chapter in Services ReferenceforAEM Forms.

To use the generatePrintedOutput operation, these items must be available to the process at run time:

- An XML file that contains the records to merge with the form design. An understanding of the XML structure is required in order to set the record level.
- An understanding of the type of output to generate. The type of output to generate depends on the print formats that the printer supports.
- The printer protocol to use, the name of the print server, and network name of the printer.

RELATED LINKS:

generatePDFOutput

sendToPrinter

## 6.9. Creating pre-filled and interactive PDF forms

This Quick Start describes how to use the Forms to render an interactive PDF form with pre-filled data. For more information about the Forms service, see Services Reference for AEM Forms.

Using the Forms service, a form design can be merged with XML data, and rendered as a pre-filled, interactive PDF form. For example, users log in to a web-based system and open an interactive purchase form using Adobe Acrobat version 8.0 or later. The form is pre-filled with shipping, contact, and purchase order information. Users fill in the remaining purchase order details and save the PDF form.

An application, named *poPDFApp*, implements this service for the organization. The form design, named *purchaseOrder.xdp*, is available as part of the application. An XML file, which contains the shipping, contact, and purchase order information, is merged with the form design to render the interactive PDF form. The resulting PDF form is saved to the network location specified by the user, such as *\\user1\POForm.pdf*. Users retrieve the PDF form from the network location, fill, save, and send it electronically.

The following assets are in the poPDFApp application:

- A form design named *purchaseOrder.xdp* that is used to render the PDF form. Form designs are created using Designer . (See *Designer Help*.)
- A process, named *createPOForm,*that includes the following items:
  - An input `document` variable, named *poShipXML*, that stores the XML data with shipping, contact, and purchase order information.
  - An input `string` variable, named *savePOLocation*, that stores the network location to save the purchase order form.
  - An output `document` variable, named *ouputPDF*, that stores the interactive PDF form.
  - A renderPDFForm operation (Forms service) that merges a form design with an XML file and renders an interactive PDF form. The resulting output is a `document` value that represents the PDF document.

- A Write Document operation (File Utilities service) that saves the `document` value to the file system.

The process diagram for the createPOForm process looks like the following illustration:

The following XML document provides the shipping information to pre-fill the rendered PDF form:

```
<?xml version="1.0" encoding="UTF-8"?>
<poform1>
<txtPONum>8745236985</txtPONum>
<dtmDate>2004-02-08</dtmDate>
<txtShipCompanyName>Fin@nce Incorporated</txtOrderedByCompanyName>
<txtShipAddress>123, Any Blvd.</txtOrderedByAddress>
<txtShipCity>Any City</txtOrderedByCity>
<txtShipStateProv>Any State</txtOrderedByStateProv>
<txtShipZipCode>12345</txtOrderedByZipCode>
<txtShipCountry>Any Country</txtOrderedByCountry>
<txtShipPhone>(123) 456-7890</txtOrderedByPhone>
<txtShipFax>(123) 456-7899</txtOrderedByFax>
<txtShipContactName>Akira Tanaka</txtOrderedByContactName>
</poform1>
```

## Configuration

For the renderPDFOrm operation, the following properties are configured to generate an interactive PDF form:

### Input

- **Form:**A literal value of poPDFApp > createPOForm/1.0 > purchaseOrder.xdp is selected to specify the form design. The Select Asset dialog box is used to select a literal value.
- **Input Data:** The poShipXML variable that contains the contents from the poShipInfo.xml file.

### PDF Options

- **Acrobat Version**: The Acrobat and Reader 8.0 and above item is selected from the list. The item specifies to generate a PDF document that requires Acrobat or Adobe Reader 8.0 or later to open it.

### Output

- **Rendered Form:** The outputPDF variable that stores the PDF form.

  *For the Write Document operation, the following properties are configured to save the interactive PDF form to a network location:*

### Input

- **Pathname Pattern:** The savePOLocation variable that specifies the name of the file.
- **Document:** The outputPDF variable that specifies the content to save as a file.

## Other considerations

Instead of writing the PDF form to a network location, consider sending the PDF form (as a byte stream) to a Java servlet. The Java servlet can display the form to a web page. (See Creating Web Applications-thatRenders Forms in *Programming with AEM Forms*.)

Consider using the processFormSubmission operation in a separate process to handle the submission of data from a rendered PDF form.

Use the Acrobat Reader DC extensions service to apply usage rights to the PDF form when users use Adobe Reader to fill the form. (See ApplyUsageRights.)

RELATED LINKS:

renderPDFForm

WriteDocument

# 6.10. Handling data submitted from a form

This Quick Start describes how to use the Forms service to create a process to use XDP data submitted from a rendered PDF form. The XDP data can be saved to a database, sent to another service for processing, or manipulated and saved for later processing. For more information about the Forms service, see Services Reference for AEM Forms.

When using the Forms service, data is submitted as XDP data. The resulting data is converted to XML data (without the XDP data). In the form design, the type of data submitted can be configured. (See *Designer Help* .) For example, an organization has a process that uses a common PDF form that is rendered to applicants to submit loan requests in North America. Separate groups in the organization handle the loan requests based on the country as follows:

- Applicants who reside in the United States have loan requests handled by Group A.
- Applicants who reside in Canada have the loan requests handled by Group B.

In a web browser, applicants fill a PDF form that is rendered by the Forms service. The country that an applicant resides in is determined by the *Country* field. Users select the country in which they reside using a drop-down list. The drop-down list contains the values *Canada* and *United States*.

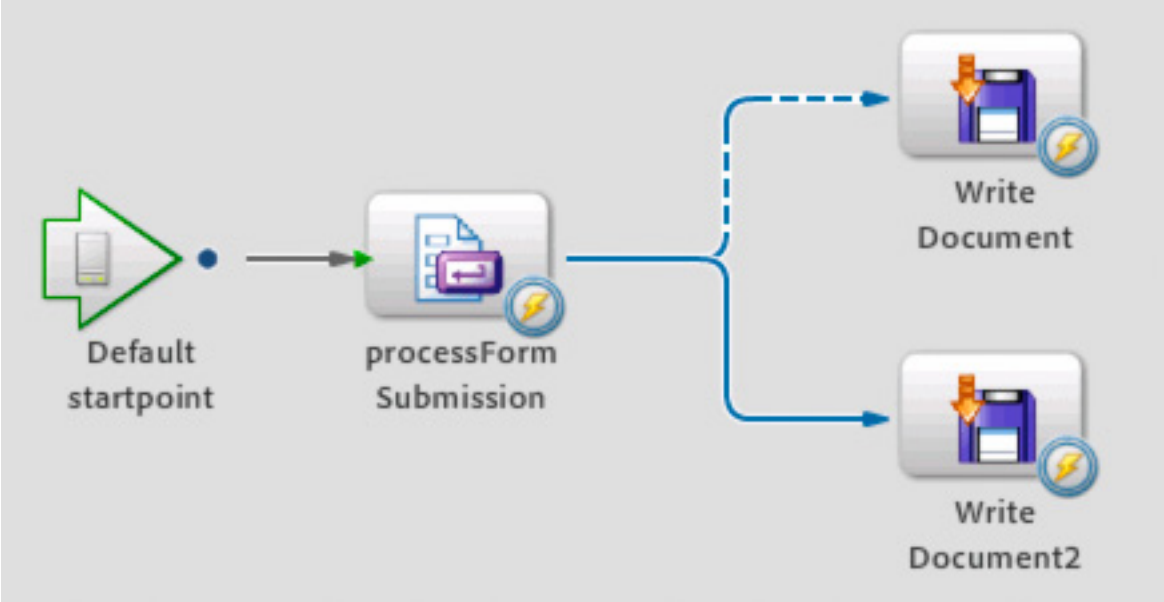
An application, named *handleSubmissionApp,* implements the service for the organization. When a user clicks the Submit button in the PDF form, XDP data is sent to a Java servlet. The servlet invokes the handleSubmissionApp service. The service uses the Forms to process the XDP data and in a separate step and saves it to an XML file. The XDP data from an applicant in the United States is saved to the network location \\GroupA_US\loan.xml. The XDP data from an applicant in Canada is saved to the network location \GROUP_B_CAN\loan.xml. To keep the loan request data files unique, a unique numeric string is appended to end of the filename. For example, loan1.xml is created if the loan.xml file exists at the network location.

The handleSubmissionApp application includes a process, named *handleFormSubmission,*that includes the following items:

- An input `document` variable, named *xdpXML,*that stores the XDP data submitted from the servlet when the Submit button is clicked.
- An output `document` variable, named *outputXML*, that stores the XML data.
- A processFormSubmission operation (Forms service) that processes the submitted data from a PDF form.
- Two Write Document operations (File Utilities service) that save the XML data as `document` values. The location to save the XML depends on whether the applicant selected Canada or United States in the drop-down list.
- A conditional route that determines the next operation to execute. When the value of the Country field is United States, the Write Document operation executes; otherwise, the Write Document2 operation executes.

The process diagram for the handleFormSubmission process looks like the following illustration:

**NOTE:** Conditional routes appear as dotted lines in the process diagrams.




The following text shows relevant parts of the XDP data for understanding this Quick Start. The data is stored in the `<xfa:datasets>` element and each element in the XML tree can be accessed in the process. In the example that follows, the applicant resides in the United States:

```
<?xml version="1.0" encoding="UTF-8"?>
<?xfa generator="XFA2_4" APIVersion="3.0.8262.0"?>
<xdp:xdp xmlns:xdp="http://ns.adobe.com/xdp/"
timeStamp="2009-11-27T19:16:38Z"
uuid="1fe71528-34a5-43df-8ad1-7f08126b3698">
...
<xfa:datasets xmlns:xfa="http://www.xfa.org/schema/xfa-data/1.0/">
<xfa:data>
<LoanApp>
<Name>Akira Tanaka</Name>
<LoanAmount>100000</LoanAmount>
<PhoneOrEmail>atanaka@sampleorganization.com</PhoneOrEmail>
<Country>United States</Country>
<ApprovalStatus>PENDING APPROVAL</ApprovalStatus>
</LoanApp>
...
<xfa:data>
...
</xdp:xdp>
```

## Configuration

For the processFormSubmission operation, the following properties are configured to retrieve information from the submitted data:

### Route Evaluation

There are routes from processFormSubmission operation to the Write Document and Write Document2 operation. Using an XPath expression, a condition is added to the route to the Write Document operation. The XPath expression evaluates the value in the Country field from the `<xfa:data>` element in the data. The following condition executes the Write Document operation when the Country field is set to United States:

- `/process_data/formData/xdp/datasets/data/LoanApp/Country = "United States"`

  **NOTE:** XPath expressions allow you access data in the XML data structure. (See CreatingXPathexpressions .

### Input

- **Submitted Form Data:** The xdpXML variable that specifies the XDP data submitted to the operation.
- **Environment Variables:**An entry where Variable Name is `CONTEXT_TYPE` and Value is`application/vnd.adobe.xdp+xml` is added. The entry specifies the type of file that is submitted.

### Form Submission Options

- **Export Data Format:** The XML Data item is selected from the list. The item specifies to export the XML data without the `<Xfa:datasets>` packaging.

### Output

- **Result Document:** The outputXML variable that contains the XDP data submitted to the operation. The XDP data is saved as XML.

  *For the Write Document operation, the following properties are configured to save XML data to a network location:*

### Input

- **Pathname Pattern:** The value of `\\GroupA_US\loan.xml` is typed. The value specifies the location and name of the PDF document.
- **Document:** The outputPDF variable that specifies the content to save as a file.
- **Make Unique**: The Append a suffix to the filename to make it unique if it has been used option is selected. The option adds a numeric string to the end of the filename, which ensures the filename is unique.

For the Write Document2 operation, the following properties are configured to save the XML data to a network location:

### Input

- **Pathname Pattern:** The value of `\\GroupB_CAN\loan.xml` is typed. The value specifies the location and name of the PDF document.
- **Document:** The outputPDF variable that specifies the content to save as a file.
- **Make Unique**: The Append a suffix to the filename to make it unique if the filename has been used option is selected. The option adds a numeric string to the end of the filename. Adding a numeric value ensures that the filename is unique.

## Other considerations

The form used to submit data must be rendered using Forms service. For example, use the Render PDF Form operation to render a PDF form for submission. You cannot use a PDF form created in Acrobat to submit a form to the Forms service. For a web-based system, consider using a Java servlet. (See Creating Web ApplicationsthatRenders Forms in *Programming with AEM Forms*.) The Java servlet can handle the rendering and the submission of the PDF form for users that use the service from a web browser.

Instead of using a Java servlet to invoke the service, consider using a web service to invoke the service. For example, JavaScript can be added to the Submit button to invoke the process. (See the Connecting to a datasource topic in *Designer Help*.)

This Quick start does not describe how to pass the XDP data from a Java servlet to invoke the service. (See Passingdatato AEM Forms services using the Java API in *Programming with AEM Forms*.)

RELATED LINKS:

processFormSubmission

WriteDocument

# 6.11. Applying usage rights to PDF documents using a watched folder

This Quick Start describes how to use the Acrobat Reader DC extensions service to apply usage rights to a PDF document. Applying usage rights to a PDF document allows Adobe Reader users to use interactive features that are available in Adobe Acrobat. Interactive features, such as commenting and the ability to save files locally, are not available in Adobe Reader. For more information about the Acrobat Reader DC extensions service, see Services Reference for AEM Forms.

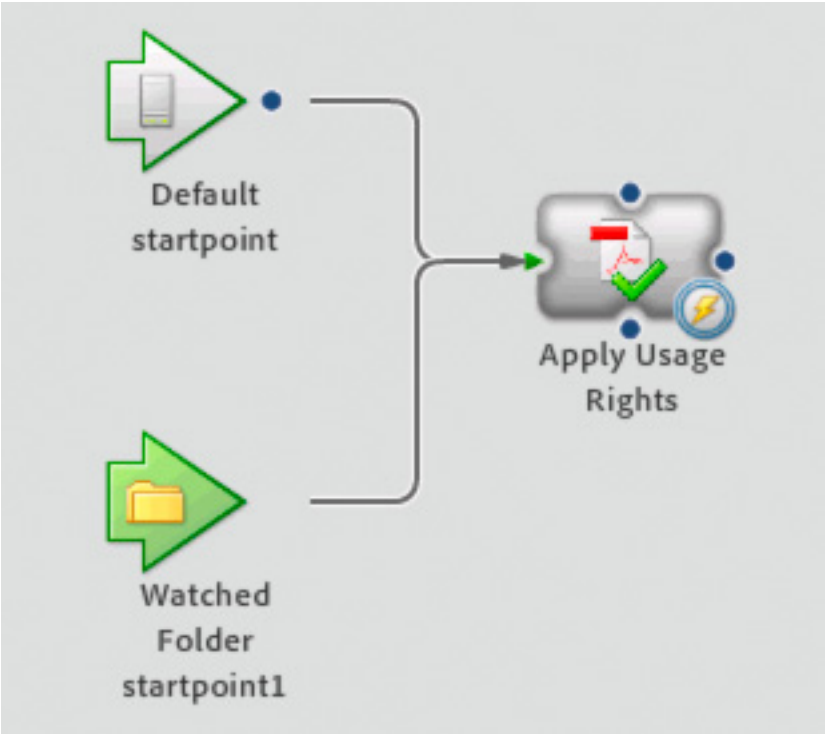
For example, an organization has a process to gather feedback about the content of PDF documents. Using AEM forms - Acrobat Reader DC extensions 10, additional usage rights can be applied to the PDF document so that users can provide comments using Adobe Reader. Users copy their PDF document to a folder and then retrieve the PDF document with usage rights applied to it from another folder.

An application, named *enablePDFApp*, implements the service for the organization. To use the service, users copy the PDF documents to a network folder named *LiveCycleES4Server**SendForReview**input*. Users retrieve the PDF document with usage rights applied to it from a network folder named *LiveCycleES4Server**SendForReview**readyForReview*. Both folders reside on the AEM Forms Server and are configured as shared folders. This configuration used to invoke the service is called a *watched folder*. A user ID of *readadmin* configured with the *Services User* role is used to invoke the service using watched folders. (See ConfiguringwatchedFolders endpoints in *AEM forms administration help*.)

The enablePDFApp application includes a process, named *applyREPDF,*that includes the following items:

- An input and output `document` variable, named *reOuputPDF*, that stores the PDF document. The process is invoked with the PDF document as input. The PDF document is fetched from the \\LiveCycleES4Server\SendForReview\input folder. The service saves the output PDF document to the \\LiveCycleES4Server\SendForReview folder.
- The credential alias, named *REEXTEND*, is used to apply the usage rights to the PDF document.
- A Watched Folder start point.
- An Apply Usage Rights operation (Acrobat Reader DC extensions service).

The process diagram for the applyREPDF process looks like the following illustration:




## Configuration

For the Apply Usage Rights operation, the following properties are configured:

### Input

- **Input PDF Document:** The reOuputPDF variable that specifies the PDF document to apply usage rights to.
- **Credential Alias:** A value of `REEXTEND` is provided. A developer or an administrator must import the credential to the server. (See Configuring credentialsforuse with Reader Extensions in *AEM Forms administration help*.)
- **Usage Rights Options:** The Commenting option is selected. The option permits users to add comments to a PDF document using Adobe Reader.
- **Reader Message:**The value `You can useAdobeReader to add comments to this document` is typed. The value specifies the message users see when they open the PDF document using Adobe Reader.

### Output

- **Output PDF Document:** The reOuputPDF variable that stores the PDF document that has usage rights applied to it.

  *For the Watched Folder start point, the following properties are configured to invoke the process:*

### General

- **Path:** A value of `\SendForReview` is typed. The value specifies the location of the watched folder on the AEM Forms Server.
- **Domain Name:**A value of `DefaultDom` is typed. The value specifies the domain name to which a user belongs. The domain for a user is configured in User Management on the AEM Forms Server. (See Setting up and managingdomains in *AEM Forms administration help.)*
- **User Name:** A value of `readmin`. The value specifies the user ID that is assigned the Server User role. The user ID must be part of the same domain configure for the Domain Name property. (See Creating and configuringroles in *AEM Forms administration help*.

### Server Configurations

- **Result Folder:** The value of `readyForReview` is typed. The value specifies the name of the folder to which to save the processed PDF document.

### Inputs/Outputs

- **Input data filter:**The value of `*.pdf` is typed. The value specifies to process only files with the PDF extension.
- **Document:** The value of `%F_review.%E` is typed. The value specifies to use the original name of the file and add the string *_review* to the end of the filename.

## Other considerations

Before using the Apply Usage Rights operation, configure a credential for applying usage rights to the PDF document on the AEM Forms Server. (See Configuring AcrobatReaderDC extensions in *AEM forms administration help*.)

When the watched folder resides on the AEM forms Server, configure it as a shared folder with proper read and write permissions for the users. Users can only use a shared folder when it is configured correctly.

RELATED LINKS:

ApplyUsage Rights

WriteDocument

# 7. Creating and managing processes

Create a *process* to represent the business process that you are automating. When you create a process, specify a name and associate it with an existing application. Processes that you create appear in the Application view.

You create processes by using the following methods:

**New AEM Forms Process wizard:**

This wizard creates a process that is either empty or contains one or more start points that are configured.

**Copy and paste:**

Copy an existing process and paste it into the same or a different application. This method is useful when you want to create a process that is similar to the one previously created. You make only the required modifications without having to redraw the entire process diagram.

**Save as:**

This method is similar to the copy-and-paste method. The only difference is that you can specify a different name for the process.

Check the newly created process into the repository to facilitate collaborative development and to preserve changes in a safe environment.

**Drag a process template:**

This method is similar to the copy-and-paste method. The only difference is that you can specify a different name for the process.

Check the newly created process into the repository to facilitate collaborative development and to preserve changes in a safe environment.

RELATED LINKS:

Process diagrams

Starting processes using start points

## 7.1. Creating processes using the New Process wizard

Use the New Process wizard to create an empty process. Optionally, you can specify the following items in the process:

- A start point for the process. A start point determines how the process is started. A process can have multiple start points. By default, a new process always contains a programmatic start point. (See Programmaticstart point properties).
- Descriptive text.

- An image that is displayed with your process in Workspace.
- Instructions for the process users that are displayed in Workspace.

## Create a process using the wizard:

1) Select File > New > Process.
2) In the Name box, provide a unique name for the process.
3) (Optional) In the Description box, type a description for the process.
4) In the Enter Or Select The Parent Folder box, specify the path to the process location within the application hierarchy. The path must include the name and version of the application and, optionally, subfolder names. An example of a valid path is /MyApp/1.0/processes. You can either type the path or select the required location from the application tree.
5) (Optional) Click New Folder to create a subfolder in the application tree. Creating subfolders in the application tree is a recommended way of organizing your assets inside the application. For example, you can group different types of processes in separate folders.
6) Click Next and, in the Configure a Start Point panel, select one of the following available start points for the process.

   ***When A User Submits A Task In Workspace:****The process starts when the user fills and submits a form in Workspace. For details about how to configure the Workspace start point, see Configuringthe Workspace start point.*

   ***When A User Submits A Task From a Mobile Device:****The process starts when the user fills and submits a form using AEM forms - Mobile 11. For details about how to configure the Mobile start point, see Configuringthe Mobile start point.*

   ***When The System Receives An Email:****The process starts when the process receives an email from the user. For details about how to configure Email start point, see Configuringthe Email start point.*

   ***When A File Arrives In A Watched Folder:****The process starts when a file is placed in a specified watched folder. For details about how to configure Watched Folder start point, see Configuringthe Watched Folder start point.*

   ***Add Start Points Later:*** *The new process has only the default programmatic start point. You can add a custom one later. (See Startingprocesses using start points).*

   **NOTE:** Selecting the Workspace or Mobile option creates a long-lived process. All other options create a short-lived process. You can change the process type later in the Process Properties dialog box. (See Short-livedprocesses and long-lived processes and Modifyingprocess properties*).*
7) Click Next. The next panel that appears depends on your selection in the Configure a Start Point panel. If you choose to add start points later, the Summary panel appears. For all other start point types, the corresponding start point configuration panel appears.
8) Carefully examine information in the Summary panel. If you must change it, click Back and modify your settings.
9) Click Finish to create a process.

## Configuring the Workspace start point

The Workspace start point configuration consists of two panels, where you provide the following information:

- (Optional) A form to use in the process. This form must already be present in your application.
- A Workspace category where the process resides.
- The name for the process as it appears in the Start Process area of Workspace.
- A process icon.
- Instructions for process users.

The first configuration panel appears after you select Workspace in the Configure a Start Point panel and click Next.

**NOTE:** You can modify the Workspace start point configuration in the Process Properties view. (See Workspacestart point properties).

**Configure the Workspace start point:**

1) In the Form Selection area, choose one of these options:

   ***Choose a Form Later:*** *Select this option if you do not have the process form ready. You can select the form later in the Process Properties view for the start point.*

   ***Use an Existing Form:****Select this option to specify a form that is stored in the application. In the space provided, specify the path to the form location within the application hierarchy. The path must include the name and version of the application and, optionally, subfolder names. An example of a valid path is /MyApp/1.0/forms/MyForm.xdp. You can either type the path or select the required location from the application tree.*

   **NOTE:** If you select a form that is submitted as an XML Data Package (XDP), the wizard creates an xml variable called formData. If you select a form that is submitted as a PDF, the wizard creates a document variable called formData.

2) Click Next and, in the Workspace Category box, choose one of these options:

   ***Choose a Workspace Category:****Select the name of the Workspace category where the process will appear. This option does not appear until there is at least one Workspace category created.*

   ***Create a New Workspace Category:*** *Type the name for the new Workspace category.*

3) (Optional) In the Workspace Process Name box, specify the name of the process as it will appear in the Start Process area of Workspace. The default name is Start <*process name*>.

4) (Optional) In the Description box, type instructions for the process user.

5) (Optional) To associate a custom image with the process, click Browse, navigate to and select an image file, and click Open. The image you specify is displayed in the process cards within Workspace. The image formats that are supported are JPG and PNG.

   **TIP:** Reduce your image to 48 x 48 pixels. You can compress the image without losing quality before you add it to your process by using tools such as Adobe ImageReady®. Compression optimizes the image for use in AEM Forms and AEM Forms app, improves load times, and maximizes user experiences. Always verify that the images are displayed correctly. Images larger than 48 x 48 pixels are scaled and may appear distorted.

6) (Optional) To disassociate a custom image from the process, click Reset To Default .
7) Click Next and, in the New Process Configuration Summary panel, examine the list of selected options. If you must change it, click Back.
8) Click Finish to create a process.

## Configuring the Mobile start point (Deprecated)

The Mobile start point (Deprecated) configuration consists of two panels, where you provide the following information:

- (Optional) A Guide to use in the process. This Guide file must already be present in your application and compatible with mobile devices.
- An Mobile category where the process resides.
- The name for the process as it appears in the mobile application.
- A process icon.
- Instructions for process users.

The first configuration panel appears after you select the Mobile option in the Configure a Start Point panel and click Next.

**NOTE:** You can modify the Mobile start point (Deprecated) configuration in the Process Properties view. (See Mobilestart point (Deprecated) properties).

**Configure the Mobile start point (Deprecated):**

1) In the Form Selection area, choose one of these options:

   ***Choose a Form Later:*** *Select this option if you do not have the Guide for the process ready. You can select the Guide later in the Process Properties view for the start point.*

   ***Use an Existing Form:****Select this option to specify a Guide that is stored in the application and is compatible with mobile devices. In the space provided, specify the path to the Guide file location within the application hierarchy. The path must include the name and version of the application and, optionally, subfolder names. An example of a valid path is /MyApp/1.0/forms/MyForm.xdp. You can either type the path or select the required location from the application tree.*

   *If the Guide you select does not support mobile devices, a warning message is displayed at the top of the panel. For more information, see Guideson mobile devices.*

   **NOTE:** If you select a Guide that is submitted as an XML Data Package (XDP), the wizard creates an xml variable called formData.
2) Click Next and, in the Mobile Category box, choose one of these options:

   ***Choose a Mobile Category:****Select the name of the Mobile category where the process is displayed. This option does not appear until there is at least one Mobile category created.*

   ***Create a New Mobile Category:*** *Type the name for the new Mobile category.*
3) (Optional) In the Mobile Process Name box, specify the name of the process as it is displayed in the Mobile application. The default name is Start <*process name*>.
4) (Optional) In the Description box, type instructions for the process user.

5) (Optional) To associate a custom image with the process, click Browse , navigate to and select an image file, and click Open. The image you specify is displayed in the Mobile application. The image formats that are supported are JPG and PNG.

   **TIP:** Reduce your image to 150 x 150 pixels. You can compress the image without losing quality before you add it to your process by using tools such as Adobe ImageReady®. Compression optimizes the image for use in AEM Forms, improves load times, and maximizes user experiences. Always verify that the images are displayed correctly. Images larger than 150x150 pixels are scaled and may appear distorted.

6) (Optional) To disassociate a custom image from the process, click Reset To Default .
7) Click Next and, in the New Process Configuration Summary panel, examine the list of selected options. If you must change it, click Back.
8) Click Finish to create a process.

## Configuring the Email start point

In the Email Configuration panel, provide the information that is required for the following operations:

- Connecting to an SMTP server to send email messages
- Connecting to a POP3 or IMAP server to receive email messages

This panel appears after you select Email in the Initial Configure a Start Point panel and click Next.

**NOTE:** You can modify the Email start point configuration in the Process Properties view. (See Emailstart point properties).

**Configure the Email start point:**

1) Provide the appropriate information in the following boxes:

   ***Inbox Host:*** *Specify the IP address or the host name of the POP3 or IMAP server to use for receiving email.*

   ***Inbox Protocol:*** *Select the POP3 or IMAP protocol to use for receiving email.*

   ***Inbox Port:****Specify the port that is used to connect to the POP3 or IMAP server.*

   ***Inbox User:*** *Specify the user name for the account to use to log in to the POP3 or IMAP server.*

   ***Inbox Password:****Specify the password that is associated with the user name specified in the Inbox User box.*

   ***SMTP Host:*** *Specify the IP address or the host name of the SMTP server to use for sending email messages.*

   ***SMTP Port:*** *Specify the port that is used to connect to the SMTP server. The default value is 25.*

   ***SMTP User:****Specify the user name for the account to use to log in to the SMTP server.*

   ***SMTP Password:****Specify the password that is associated with the user name specified in the SMTP User box.*

2) Click Next and, in the New Process Configuration Summary panel, examine the list of selected options. If you must change it, click Back.
3) Click Finish to create a process.

### Configuring the Watched Folder start point

In the Watched Folder Configuration panel, specify the watched folder location on the AEM Forms Server and the filename pattern to match before starting the process.

A *watched folder* is a folder that is located on the AEM forms Server computer. The default path is /WatchedFolders/application name/process name. These folders are created in the root of the server computer when you deploy the application. For example, if your application name is *myapplication* and your process name is *myprocess*, AEM Forms creates the folders C:/WatchedFolders/myapplication/myprocess. If the watched folder exists on the server, specify its path instead. In a clustered environment, the path must point to a shared network folder that is accessible from every computer in the cluster. (See "Using watched folders" in *administration console Help)*.

The filename pattern can use the asterisk (*) and question mark (?) symbols:

- Use the asterisk (*) to represent any series of characters. For example, the pattern *te*.xdp* matches the filename test.xdp.
- Use the question mark (?) to represent any single character. For example, the pattern *te?t.xdp* matches the filename test.xdp.

The Watched Folder Configuration panel appears after you select Watched Folder in the Configure a Start Point panel and click Next.

**NOTE:** You can modify the Watched Folder start point configuration in the Process Properties view. (See Watchedfolder start point properties).

**Configure the Watched Folder start point:**

1) In the Path box, specify the location of the watched folder or accept the default location.
2) (Optional) In the Include File Pattern box, specify the filename pattern so that only the files that match this pattern will start a process. The default pattern is the asterisk (*), which means that any file that is placed in the watched folder starts the process.
3) Click Next and, in the New Process Configuration Summary panel, examine the list of selected options. If you must change it, click Back.
4) Click Finish to create a process.

RELATED LINKS:

Creating processes based on existing process

Modifying processes

## 7.2. Process editing aids

You can simplify the drawing of your process diagram by performing these tasks:

- Copy parts of the process diagram.
- Modify the magnification of the process diagram.

## Copying elements

You can copy any element or multiple elements that include operations, routes, events, gateway elements, activity elements, and annotation elements from your process diagram to the same process or to another process.

**NOTE:** If your selection includes a start point event and you try to copy it to the same process diagram, everything will be copied except the start point event. This is because you can only have one start point of an event type in each process diagram. If you want to copy a branch, see Adding,editing, deleting, and copying branches.

**Copy an element:**

1) In the process diagram, select the element to copy.
2) Perform one of the following actions to copy and paste an element:
   – In the toolbar, click Copy and then click Paste .
   – Select Edit > Copy and then Edit > Paste.
3) (Optional) Click the process editor tab of the process diagram where the element is to be copied.
4) In the process diagram, click the location where the element is to be copied.

   **TIP:** You can select multiple elements using one of the following methods:
   – Hold down the Control key and select each element you want to copy.
   – Draw a border around the elements you want to copy.

## Changing the magnification of process diagrams

Use the zoom tool to increase or decrease the magnification of a process diagram.

- Click the plus sign button to increase the magnification.
- Click the minus sign button to decrease the magnification.
- Drag the slider or click on the scale to increase or decrease the magnification.

## Variable highlighting

You can click on an activity in the process editor, to see the variables that are referenced by that activity displayed in blue text in the Variable view. Similarly, you can select a variable in the Variable view, to see the activities that are reference by that variable highlighted in blue in the process editor.

RELATED LINKS:

Working with operations

Process diagrams

# 7.3. Adding annotations

Add an annotation element to the visual representation to convey information about the process diagram. For example, to easily identify the process diagram when it is opened in the process editor, use the annotation element as a title for the process diagram.

When you add an annotation element, you can also specify the font properties. You can modify the annotation element properties or delete the annotation element at any time.

Annotation elements do not affect the behavior of processes at run time. Also, the properties or text of annotation elements cannot be modified at run time.

## To add an annotation element:

1) Drag the Annotation modelling element to the process diagram.

## To modify the annotation element:

1) Double-click the annotation element, or right-click it and select Annotation Properties.
2) In the Text box, type the text you want to appear in the annotation element.
3) (Optional) In the Font list, select the font you want to use for the annotation element. The fonts that are available depend on which fonts are installed on your computer.
4) (Optional) In the Size list, select a size for the text.
5) (Optional) Specify whether you want the annotation element to have a border:
   – To show a border, select Border.
   – To show no border, clear Border.
6) (Optional) Specify whether you want the background color of the annotation to be different than the swimlane where it is located:
   – To specify a background color, select Background Color, click the ellipsis button , and select the color.
7) Click OK.

## To delete an annotation element:

1) In the process diagram, perform one of the following tasks to delete the annotation element:
   – Right-click the annotation element and select Delete Annotation.
   – Select the annotation element and in the toolbar, click Delete , or press the Delete key, or select Edit > Delete.

RELATED LINKS:

Organizing operations in swimlanes

# 7.4. Creating processes based on existing process

Another way of creating a process is to start with a copy of an existing process and then modify it as required. This approach is useful when you want to create a process that is similar to the one you already have. You can also use this approach when you want to experiment with variations of the same process.

You can create a copy of the process either by using copy and paste or by using the Save As command. When using the Save As command, you can specify the new name for the copied process. The copy and paste method prompts for the new name only when the process is copied into the same application. If the process is copied to a different application, it retains its original name, which cannot be changed.

## Create a process using copy and paste:

1) In the Applications view, right-click the process and select Copy.
2) Right-click the destination location and select Paste:
   – To paste the process at the root of the application version, right-click the application version node.
   – To paste the process in a folder under the application version, right-click the folder name.
3) If you paste the process into the same application version, in the Name Conflict dialog box, type a new name.

## Create a process using Save As:

1) In the Applications view, select a process, and then select File > Save As. In the Save As dialog box, the description is copied from the process you selected.
2) In the Name box, type the name for the new process.
3) (Optional) In the Description box, type text to replace the existing description.
4) In the Enter Or Select The Parent Folder box, specify the path to the process location within the application hierarchy. The path must include the name and version of the application and, optionally, subfolder names. An example of a valid path is /MyApp/1.0/processes. You can either type the path or select the required location from the application tree.
5) (Optional) Click New Folder to create a subfolder in the application tree. Creating subfolders in the application tree is a recommended way of organizing your assets inside the application. For example, you can group different types of processes into separate folders.
6) Click Finish.

RELATED LINKS:

Creating processes using the New Process wizard

Modifying processes

# 7.5. Managing processes

After you finish designing your process, you can use the Applications view to manage the process.

**NOTE:** Use the Processes view to manage processes created in LiveCycle ES (8.x). Use the Applications view to manage processes created in LiveCycle ES2 (9.x).

## Opening and closing processes

Open a process to view or edit from the Applications view. (See Editing and viewing assets.) When you open a process, the associated process diagram opens in a process editor. You can only open processes that are in your local view.

What you can do with a process depends on whether it is checked out (See Checking out applications or assets.):

- When the process is not checked out, you cannot save modifications to the process variables, process diagram, or process properties. You can save the process with a different name. It is possible for other developers to edit a process while you are viewing it.
- When the process is checked out, you can save modifications to the process variables, process diagram, and process properties. The process is locked, and other users can change it only when the force a checkout.

**Open the process:**

1) In the Applications view, right-click the name of the process and select Open.

**Close the process:**

1) In the process editor, click Close. If you did not saved the process, you are prompted to save it.

## Saving processes

When you save the process, the changes are saved to the local disk only. To save process changes to the server, check the process into the repository. (See Checkingin applications or assets.)

Under the following circumstances, you are prompted to update configuration parameter properties on the AEM Forms Server:

- The application containing the process is deployed.
- The process includes one or more configuration parameters whose default values are different from the server values.

Under these circumstances, when you save the process, you need to indicate whether to update the server with the current default values.

When you update the properties on the server, the following changes are made:

- Parameter types and default values are changed to the types and values that are configured in the Variables view.

- New configuration parameters are added to the server
- Deleted parameters are removed from the server.

**Save the process to local disk:**

1) In the toolbar, click Save .
2) If a Warning dialog box appears, specify whether to update the configuration parameter properties on the AEM Forms Server:
   – To update the properties, select Replace Configuration Parameter Values on the AEM Forms Server.
   – To keep the properties on the server unchanged, deselect Replace Configuration Parameter Values on the AEM Forms Server.

RELATED LINKS:

About Processes

Creating processes using the New Process wizard

Creating processes based on existing process

# 7.6. Modifying processes

You can modify an existing process. First check out the process, then open it and modify as required. You can also modify the properties of your process. For example, you can modify the image associated with the process version, transaction settings, or registered prefixes for namespaces and XPath expressions in the process. Processes that are no longer required can be deleted from the application.

You can modify a process that was invoked and has process instances associated with it. When you change a deployed process, the changes are implemented after you redeploy the application. Existing and new process instances associated with the process will progress according to the latest changes. Use caution when changing an activated process. Some changes can cause existing process instances to stall. For example, if you remove variables that are referenced in XPath expressions, when the XPath expression is executed, the process instance will stall.

## Change process properties

Change process properties in the Configure Process dialog box. Before you open this dialog box, check out the process to make the fields writable. If you open the Configure Process dialog box for a process that is not checked out, you can view the properties of the process but you cannot change them.

**TIP:** *If you do not check out a process before you start editing it, Workbench will prompt you to check it out once you start making changes in the process.*

**Change process properties:**

1) In the Applications view, right-click the process name and select Check Out.
2) Right-click the process name again and select Configure.
3) On the General tab, specify or change the following information, as required:

***Title:*** *The title of the process as it appears in the Start Process area of Workspace.*

***Description:*** *A short text that describes the process.*

***Process Image:*** *The image that is associated with the process. This image is displayed in the process cards within Workspace. The image formats that are supported are JPG and PNG. Click the Reset to Default button to disassociate an image from the process. Click the Browse button to navigate to and select an image file, and then click Open.*

4) Click the Advanced tab.

5) (Optional) In the Type list, select one of these options:

***long-lived****: Specifies to store information about process instances in the database.*

***short-lived****: Specifies to not store information about process instances in the database. Processes are short-lived by default. (See Short-livedprocesses and long-lived processes.)*

6) (Optional) In the Propagation list, select one of the following propagation contexts to specify how a transaction context is propagated:

**NOTE:** The Propagation list is available only if you selected short-lived in step 5.

***MANDATORY****: Supports a transaction context if one exists. An exception occurs if a transaction context does not exist.*

***NEVER****: Always executes without a transaction context. An exception occurs if an active transaction context exists.*

***NOT SUPPORTED****: Does not support execution with an active transaction context. Always executes without a transaction context. If an active transaction context exists, it is suspended.*

***REQUIRED****: Supports a transaction context if one exists; otherwise, a new transaction context is created.*

***REQUIRES NEW****: Always creates a transaction context. If an active transaction context exists, it is suspended. This value is the default.*

***SUPPORTS****: Supports a transaction context if one exists; otherwise, executes without a transaction context.*

7) (Optional) In the Timeout area, which is available only if you selected short-lived in step 5, select one of these time-out options:

***Default****: Specifies to use the application server's transaction time-out setting.*

***Number Of Seconds****: Specifies to wait the number of seconds specified before rolling back a transaction.*

8) (Optional) To add a new registered prefix to a namespace URI, perform these tasks (see Settingup templates for event data and messages):
   - In the Registered Prefixes For Namespaces Used In XPaths area, click Add A List Entry.
   - In the Prefix column, select *[Prefix]* and type a prefix.
   - In the NamespaceURI column, select *[Namespace]* and type the name of the namespace URI.

9) (Optional) To remove an existing registered prefix that is assigned to a namespace URI, select the prefix and click Delete Selected List Entry.

10) Click OK and then select File >Save to save the properties of the process.

# 7.7. Deleting processes

You can delete processes that you no longer require after they are checked in. Because all processes are visible to others who use the AEM Forms Server, exercise caution before you delete a process. Otherwise, the following situations may occur:

- Processes that use the deleted process as a subprocess will no longer function.
- Process instances of a deleted process will stall.

You can delete processes in the Processes view and in the Applications view. Use the Processes view to delete processes created in LiveCycle ES (8.x). Use the Applications view to delete processes created in AEM Forms Server. Undeploy the application before deleting a process. (See Deploying applications.)

You can delete processes that are locked if you are assigned the Application Administrator or Process Administrator role. It is recommended that you consult with the developer who locked the process to ensure that you can safely delete the process.

**NOTE:** Deleting processes that were imported as an archive file does not remove associated assets automatically. To remove a process and all of its assets, remove the archive file by using Applications and Services in administration console. (See AEM Forms administration help.)

### Delete a process:

1) In the Processes view, right-click the process and select Delete.
2) If a message appears prompting you to confirm the deletion, click Yes.

# 7.8. Activating and deactivating LiveCycle ES (8.x) processes

When you deploy an application that contains a process, the process becomes a run time service that is active and ready to be used with client applications such as Workspace or other processes. (See Deploying applications.)

A process can have multiple active process versions associated with the deployed versions of the application. If you activate a more recent process version, subsequent new process instances will use the most recent version. For example, if you activate process version 1.0 and then activate 2.0, version 2.0 will be used to create new process instances. However, if you activate process version 2.0 and then activate 1.0, process version 2.0 will still be used to create new process instances.

You can invoke specific process versions only by using the AEM forms SDK.

Although you can modify an active process, remember that changes can cause running process instances to stall. You must restart the process instances by using forms workflow from Administration Console. (See AEM Forms administration help.)

Deactivate processes to make them unavailable to other processes and client applications for invoking new process instances.

**IMPORTANT:** Deactivating a process that has outstanding process instances causes the process instances to stall.

### Deactivate a process:

1) In the Processes view, right-click the process version and select Deactivate.

### Activate a process:

1) In the Processes view, right-click the process version and select Activate.

## 7.9. Exporting and importing LiveCycle ES (8.x) processes

Export and import processes created in LiveCycle ES (8.x) for creating backup copies and sharing existing work.

Export a process and import a process to move only the process to another server. The process is exported in XML format. The file with extension .process contains only the process definition. Associated assets must be manually exported and imported.

It is recommended that you move your processes by creating a AEM Forms archive file. However, you may want to export and import the process for the following reasons:

- To export and import processes that are not activated.
- To move only the process, not the associated assets.

### Exporting processes

Exporting a LiveCycle ES (8.x) process creates a process definition file that contains electronic definitions of business processes and the information about how the business process is automated. After you save the process to a file, you can move it to another computer.

The name of the export file uses the pattern *[process name]*.process, where *process name* is the name of the process that you are exporting.

**Export a process:**

1) Select Window > Show View > Processes.
2) Right-click the process and select Export Process.
3) In the Export Process As dialog box, in the File Name box, type a name or accept the default name.
4) Select the folder to save the export file, and then click Save.

### Importing processes

You can import a process from a process definition file. The environment that you import a process into must be similar to the environment that you exported it from so that the properties in the process version are valid. Therefore, before you import a process, perform the following tasks:

- Ensure that the components and services that the process uses are available.

- Ensure that the expressions used in operation properties reference valid assets.
- Ensure that the subprocesses used are activated and running.

**IMPORTANT:** T*he JDBC, FTP, email, LDAP, group and user, and data source settings may be different in the environment into which you import the process. For example, you want to import a process that includes AssignTask operations that assign tasks to users or groups who do not exist in your environment. You may need to change the AssignTask operation properties to make the process function within the environment into which you import the process.*

**Import a process:**

1) Select File > Import.
2) Click General > File System and then click Next.
3) Click Browse and select the folder that contains the process definition file.
4) In the right pane, click the box beside the process definition file. You can select multiple files.

   **TIP:** To see only certain file types, click Filter Types and select those file types.
5) Click Browse beside the Into Folder box and select the application version into which you want to import the process.
6) To replace the files in the application that have the same name and location as the items that you are importing, select Overwrite Existing Resources Without Warning. If you do not select this option, decide whether to replace same-named files when they are being imported.
7) Click Finish.

# 7.10. Grouping in Processes

***New for 10***

## Grouping activities in your process

You can group logically and functionally coherent activities of a process using the grouping feature. These activities are visually collected and can be collapsed on the process designer as a single activity. You can also expand the group to modify any member activities.

Activities present on the same swimlane in your process diagram can be grouped.

1) Select activities you want to group. Right click any of the selected activities and click Group.
2) On the Group Properties panel, provide a meaningful name and description for the Group. Also choose a color, to help visually represent the Group. Click OK.

***NOTE:*** *To ungroup, select the group whose activities you want to ungroup. Right click the group and click Ungroup. This restores the activities to their original states.*

# 7.11. Looping in Processes

***New for 10***

In earlier versions of Workbench, users had to manually add circular routes with conditions to implement looping. In addition, counters were created explicitly to track execution and termination of the loop.

The Iteration Wizard feature in Workbench allows you to automatically generate loops based on subject, size, and data source.

Adding a loop is a two-step process:

- **Identify a data source**: The data source for the loop can be an XML or a collection variable (List or Map).
- **Identify the subject**: The subject for the loop is an empty subprocess, which can be defined with a service operation at a later point.

## Adding a loop

1) In the process designer, drag the Iteration Wizard onto a swimlane in your process diagram.
2) On the Iteration Wizard, identify the data source the loop can iterate on. Use one of the following variable types:
   - **XML variable**: Choose the XML Variable option and select your pre-defined XML variable from the drop-down list. You can also choose to add an XML variable by clicking the button. Use the Select an Element from The XML Variable field to select an element from your XML variable. Click the button to open XPath builder and build an XPath expression as required, which is substituted for the element. The loop iterates over all repetitive instances of this element.
   - **Collection variable**: Choose the Collection Variable (List or Map) option and select your pre-defined List or Map variable from the drop-down list. You can also choose to add a new variable by clicking the button.

   *Use the Maximum Loop Size option to specify the maximum number of iterations for the loop. Click **Next**.*
3) Enter a name for the new empty subprocess that handles each element of the loop. Click **Next**.
4) Review the summary of the setup you completed. Click **Finish**.

## Understanding the loop

Loops in Workbench consist of the following contained in a collapsible group:

- An empty subprocess that is designed to handle each element of the loop. Define the subprocess activity with a desired service operation for the loop to execute iteratively.
- A set of three auto-generated variables:
  - **iteration_wizard_datasize**: Reads the number of elements in the XML variable or represents the size of the collection variable.

  - **iteration_wizard_data**: Stores the current value of the data source that the loop executes each time it is run.
  - **iteration_wizard_counter**: Stores the current iteration count of the loop. Starting with zero, the value is incremented each time the loop executes successfully.
  - **iteration_wizard_valueList**: Generated only when the data source of the loop is a Map variable. This List variable stores all values of the Map.

- Two decision points evaluate further execution of the loop using two distinct routes:
  - The *exit infinite loop* route is followed when the iteration_wizard_counter exceeds the maximum size of the loop.
  - The *has next element* route is followed when the validation for *exit infinite loop* route fails. Indicates that the loop still has elements in its XML variable or collection left to be executed.

# 8. Starting processes using start points

Start points represent the invocation of a process and appear as the first item on a process diagram.

Start points enable the use of the following tools to invoke processes:

**Workspace:**

Users open a form or Guide from the Start Page to create a task. The process is invoked when users submit the task. Typically, Workspace start points are used to invoke human-centric processes.

**Mobile device:**

Users open a Guide from the AEM Forms Mobile applications to create a task. The process is invoked when users submit the task. Typically, AEM Forms Mobile start points (Deprecated) are used to invoke human-centric processes.

**Email:**

Users send an email to the AEM Forms Server to invoke the process. When processes require document values as input, the values can be provided as file attachments. String, number, and Boolean values can also be specified upon invocation. Other complex, list, or map values cannot be provided using email.

**Watched folders:**

Users copy a file or folder to a watched folder to invoke the process:

- Each file that is copied to the watched folder invokes a process. The file is used as the input for a document variable.
- Each folder that is copied to the watched folder invokes a process. Each file in the folder is mapped to a document value that is required as input.

  **NOTE:** String, number, and Boolean values can also be specified upon invocation. Other complex, list, or map values cannot be provided using watched folders.

**Custom AEM Forms client software:**

Java and web service applications, as well as applications built with Flex, can invoke processes. Custom client software can provide the value of any data type as input to the processes they invoke. Programmatic start points enable client software to invoke processes.

When a process does not include a start point, only other processes can invoke it.

Start points on a process diagram are equivalent to the endpoints of a service that is part of AEM Forms. When a process is deployed, the AEM Forms Server creates a service for the process. The

server also creates endpoints for the service for the start points that the process contains. The endpoints enable the service to be invoked.

Endpoints can also be created using administration console:

- Endpoints that are created using administration console do not appear as start points on process diagrams.
- Changes to start points in the process do not affect endpoints that are created using administration console.
- Endpoints that are created from process start points can be modified using administration console. However, if the process is redeployed, the start point configuration overwrites the configuration that was applied using administration console.

You can use the New Process wizard to add a start point when you first create the process. Use the following procedure to add start points to the process diagram. You can add as many start points as needed. After you add the start point, configure it.

**NOTE:** Events can also be configured as start points, which listen for events that can cause a process to start.

# 8.1. Add and configure a start point:

1) Drag the start point icon onto the process diagram.
2) Select the type of start point to add and click OK.
3) Click the start point and provide values for properties in the Process Properties view.

RELATED LINKS:

Event start points

# 8.2. Workspace start point properties

The properties of Workspace start points control many aspects of the user experience in Workspace:

- The information about the process that appears on the Workspace Start Process page
- The presentation asset and data that are displayed to users
- The Workspace features that can be used

When the process is deployed, Workspace start points cause TaskManager endpoints to be created on the AEM Forms Server. The administration console provides additional properties that you can configure for TaskManager endpoints.

## General

Information that appears on the Workspace Start Process page. You can also provide information for design-time tracking.

**Name:**

The name of the start point. The name appears on the Start Process page to represent the process. Provide a name that is meaningful to Workspace users.

**Description:**

(Optional) A description of the process. The description appears on the process card on the Start Process area of Workspace . Provide a description that is meaningful to users. For information about how to format the text using HTML, see Taskinstructions on task cards.

**Category:**

The category to include the process in on the Start Process page. Create a new category or select an existing one:

- To create a new category, select Create On Deploy and provide the name and description of the category. The category is created when the process is deployed with the application.
- To use an existing category, select Use Existing Category and select the category from the list.

**Contact Info:**

(Optional) Information about what to do when issues around the process occur. For example, you can provide the name and email address of a person in your organization who should be notified. The text you provide appears in Workbench, as well as in the end point properties in administration console. Notification does not occur automatically.

**Task Instructions:**

(Optional) Instructions that inform users about the functionality of the process and how to use it. Instructions appear in the following areas of Workspace:

- The Task Details tab, after the process card is opened and the task appears.
- After the process card is opened, the task is saved as a draft, and then the draft is opened.

For information about how to format the text using HTML, see Taskinstructions on task cards.

## Presentation & Data

Properties for displaying forms or Guides. (See Designingdata capture and presentation.)

**Use an application asset**

Select to display an application asset, such as a form or Guide, to the user.

**Use a CRX asset**

Select to display an adaptive form, adaptive document, or form set.

**Asset:**

The presentation asset to display to the user. Click the ellipsis button and select a form or Guide file. You can select a file from any application.

**Action Profile:**

The action profile to use with the asset. The action profiles that appear are already created for the asset that you selected. For more information, see Aboutaction profiles.

## Start Point Output

Property for saving submitted data.

**Variable:**

The variable in which to store the submitted data. Select either an xml, document, or Task Result variable:

- Select an xml variable if the asset submits field data that is XML or XDP format.
- Select a document variable if the asset is a PDF form that submits a PDF document.
- Select a Task Result variable to save the field data as well as other information about the task.

For more information about assets and the type of data they generate, see About captureddata.

**Reader Submit**

Properties for indicating Adobe Reader users start the process. These properties are available only when a PDF or XDP form is used as the value of the Asset property.

**Submit Via Reader:**

Select when users use Adobe Reader to open PDF forms. This property causes submit buttons to appear in Workspace when Adobe Reader is used.

**Submit As:**

The type of information to submit. Select XDP to submit XDP data, or PDF to submit a PDF document.

## Options

Properties for indicating the Workspace features that are available to users.

**Visible in Mobile Workspace:**

Select this option to ensure that the selected startpoint is displayed to users with "Services User" role in the AEM Forms Mobile Workspace app. For details on using start points in the Mobile Workspace app, see Working with Mobile Startpoints.

**User Can Forward Task:**

Select to enable users to forward the task to another user.

**Task Initially Locked:**

For users who have shared their queue, select to lock the task automatically when users open the task. When the task is locked, users who can access the shared queue cannot open the task.

**Add ACLs For Shared Queues:**

Select to apply the properties selected in the Options property group for users who can access tasks in shared queues.

### Attachment Options

**Permit adding attachments:**

Select to enable users to attach files to the task.

**Save task attachments in a process variable:**

Select a variable that contains files to attach to the task. The variable must be a list of document values. Click the plus sign button to create a variable for saving the files.

### Workspace User Interface

Properties for specifying the user interface to be used with Workspace, and how that interface should display when a task is opened.

**Custom:**

Select this option to customize Workspace interface, when displaying tasks. The custom interface is a swc file.

**Maximize The Form When Opened In Workspace:**

Select to display the asset using all the available space in the web browser window. Whether this option is selected or not, users can maximize or minimize the display area of the asset as needed.

## 8.3. Mobile start point (Deprecated) properties

The properties of Mobile start points (Deprecated) control many aspects of the user experience in Mobile:

- The information about the process that appears in the Mobile application.
- The presentation asset (Guide) and data that are displayed to users
- The AEM Forms features that can be used

When the process is deployed, Mobile start points (Deprecated) cause Mobile start points (Deprecated) to be created on the AEM forms Server. The Administration Console provides additional properties that you can configure for TaskManager endpoints.

## General

Information that appears in the Mobile application. You can also provide information for design-time tracking.

**Name:**

The name of the start point. Provide a name that is meaningful to users.

**Description:**

(Optional) A description of the process. Provide a description that is meaningful to users. For information about how to format the text using HTML, see Taskinstructions on task cards.

**Category:**

The Forms screen category in which the process is displayed in the Mobile Application. (Categories are not displayed for AEM forms-Mobile for Android). Create a category or select an existing one:

- To create a category, select Create On Deploy and provide the name and description of the category. The category is created when the process is deployed with the application.
- To use an existing category, select Use Existing Category and select the category from the list.

**Contact Info:**

(Optional) Information about what to do when issues around the process occur. For example, you can provide the name and email address of a person in your organization to notify. The text you provide appears in Workbench, as well as in the end point properties in administration console. Notification does not occur automatically.

**Task Instructions:**

(Optional) Instructions that inform users about the functionality of the process and how to use it. The instructions are displayed in the Task Details screen of the application.

For information about how to format the text using HTML, see Taskinstructions on task cards.

## Presentation & Data

Properties for displaying Guides. (See Designingdata capture and presentation.)

**Asset:**

The presentation asset to display to the user. Click the ellipsis button and select a Guide file that is compatible with mobile devices. You can select a file from any application.

If the form you select does not support mobile devices, a warning message is displayed in the dialog box. For more information, see Guideson mobile devices.

**Action Profile:**

The action profile to use with the Guide. The action profiles that appear are already created for the Guide that you selected. To render and submit the Guide from mobile devices correctly, ensure that the render process and submit process are selected in the profile.

### Start Point Output

Property for saving submitted data.

**Variable:**

The variable in which to store the submitted data. Select either an xml or Task Result variable:

- Select an xml variable if the asset submits field data that is in XML or XDP format.
- Select a Task Result variable to save the field data as well as other information about the task.

For more information about assets and the type of data they generate, see About capturedata.

### Options

Properties for indicating the features that are available to users.

**User Can Forward Task:**

Select to enable users to forward the task to another user.

**Task Initially Locked:**

For users who have shared their queue, select to lock the task automatically when users open the task. When the task is locked, users who can access the shared queue cannot open the task.

**Add ACLs For Shared Queues:**

Select to apply the selected Options properties for users who can access tasks in shared queues.

## 8.4. Email start point properties

Use the Process Properties view to configure the following Email start point properties:

### General

Properties for identifying the start point and determining how the process is invoked.

**Name:**

The name of the start point. The name appears in the Service Management pages of administration console. Provide a name that is meaningful to both Workbench developers and service administrators.

**Description:**

(Optional) A description of the start point. Provide a description for other developers who edit the process. The text you provide also appears as a description for the corresponding endpoint in the Service Management pages of administration console.

**Invoke Asynchronously:**

Determines the number of responses that are sent when a received email includes multiple attachments. (The process is invoked once for each attachment.) When selected, a response is returned for each invocation. When not selected, a single response is returned that includes the output for all process invocations.

For example, an email endpoint is created for a service that requires a single Word document as input and returns it as a PDF file. An email that includes three Word files is sent to the inbox. When the endpoint is configured as synchronous, a single response email is sent that includes three PDF files. If the endpoint is configured as asynchronous, three response emails are sent. Each email includes a single PDF attachment.

By default, this option is selected (asynchronous invocation).

**Domain Name:**

The User Management domain of the user that is used to invoke the process. The default value is DefaultDom, the default domain that is created for AEM Forms.

**User Name:**

The name of the AEM Forms to invoke the process. The user must be assigned the Services User role. This value is the user ID property, which is the name used to log in to AEM Forms. The default value is SuperAdmin.

## Options

Properties for configuring how users interact with the AEM Forms Server using email. You also specify how the AEM Forms Server interprets file attachments.

**Successful Job's Recipients:**

An email address to which messages are sent to indicate the process completed successfully:

- The default value is sender, which sends messages to the reply-to address of the email that invoked the process.
- You can specify a maximum of 100 email addresses. Separate multiple addresses with commas (,).
- To send no email message, provide no value.

**Failed Job's Recipients:**

An email address to which messages are sent to indicate that the process did not complete successfully:

- The default value is sender, which sends messages to the reply-to address of the email that invoked the process.
- You can specify a maximum of 100 email addresses. Separate multiple addresses with commas (,).
- To send no email message, provide no value.

**Character Set Encoding:**

The character set that AEM Forms Server uses to encode email attachments. This value must match the character set that was used to create the attachments.

**Failed Email Sent Folder:**

The email folder in which to store email messages that the AEM Forms Server could not send to the SMTP server. For example, email cannot be sent if the SMTP server is not operational.

**Domain Pattern:**

A pattern that matches the domain from which email messages are accepted. The AEM Forms Server ignores email from domains that do not match the pattern. For example, for the pattern adobe.com, only email from the adobe.com domain can invoke the process.

**File Pattern:**

A pattern that matches the file names of attachments that are used as input to the process. The value is case-sensitive. The following examples illustrate valid values:

- `*.pdf` uses all files with the file name extension .pdf.
- `data` uses all files named data.
- `*.[dD][aA][Tt]` uses all files with the file name extension .dat regardless of letter case. For example, both of the files named file1.Dat and file2.dAt are used.

## Server Configurations

Properties that enable the AEM Forms Server to connect to the email server. An email account that the AEM Forms Server uses must be configured on the email server. Contact your email server administrator for the correct values to use.

**Inbox Host:**

The name or IP address of the email server that hosts the inbox where the AEM Forms Server receives email. The default value is `localhost`.

**Inbox Protocol:**

The email protocol that the email server uses. Select either POP3 or IMAP. The default value is IMAP.

**Inbox Port:**

The port that the email server uses. The default value for POP3 is 110, and the default value for IMAP is 143.

**Inbox Time Out:**

The amount of time that the AEM Forms Server waits while checking for new email messages before a time-out occurs. This value is in seconds. Specify a value that is large enough to provide enough time to receive a response from the email server when it is running. Large values hinder server productivity when the email server is offline. The default value is `60`.

**Inbox User:**

The user name that the AEM Forms Server uses to log in to the POP3 or IMAP server. Depending on the email server and configuration, provide only the user name portion of the email or the full email address.

**Inbox Password:**

The password for the POP3 or IMAP user account.

**SMTP Host:**

The name or IP address of the SMTP server. The AEM Forms Server uses the SMTP host to send email messages.

**SMTP Port:**

The port that the SMTP server users. The default value is 25.

**SMTP User:**

The user name that the AEM Forms Server uses to log in to the SMTP server.

**SMTP Password:**

The password for the SMTP user account.

**Send From:**

The email address (for example, user@company.com) used to send email notifications of results and errors.

If you do not specify a Send From value, the email server attempts to determine the email address automatically. The server combines the value of the SMTP User property with the default domain that is configured on the email server. If your email server does not have a default domain and you do not specify a value for Send From, errors can occur. To ensure that the email messages have the correct from address, specify a value for the Send From setting.

**SMTP SSL Enabled:**

Select to force the AEM Forms Server to use SSL to connect to the SMTP server. Ensure that the SMTP server supports SSL.

**POP3/IMAP SSL Enabled:**

Select to force the AEM Forms Server to use SSL to connect to the POP3 or SMTP server. Ensure that the POP3 or IMAP server supports SSL.

## Scheduling

Properties that specify how often the AEM Forms Server checks for new email messages. You can also specify the maximum number of messages the server processes each time it checks for new email messages. Typically, the default values are adequate.

The configuration of Scheduling properties can affect the productivity of the AEM Forms Server. The values you provide depend on the expected frequency of new email messages. For example if you expect to receive one message every 10 minutes, server resources are wasted if email is checked every 10 seconds.

If many email messages are received at once, the AEM Forms Server can be processing messages when it is scheduled to check for messages again. To avoid this situation, specify a small batch size to minimize the time required to process email at each interval.

**Cron Expression:**

A cron expression that schedules when the AEM Forms Server checks for new email messages. Specify a value only if the POP3 or IMAP server requires one. Contact the POP3 or IMAP server administrator for information about how to formulate the cron expression.

**Repeat Interval**

The time frame to wait for performing the next scan for email. The value provided is in seconds. The default value is 10.

**Repeat Count:**

The number of times the AEM Forms Server scans the inbox. The value -1 indicates indefinite scanning. The default value is -1.

**Batch Size:**

The number of emails the receiver processes per scan for optimum performance. A value of `-1` indicates all emails. The default value is `2`.

**Delay When Job Starts:**

The time to wait to check for email after the scheduled time.The default value is 0.

## Inputs/Outputs

Properties for specifying process input values and for storing process output values.

**Input**

Specify a value for each input variable of the process. The type of variable determines how you express the value:

**Simple data types:**

Provide literal values for the input value of simple data types such as string, number-based types, xml, and date and time-based types. For example, the text `SomeText` is provided as a string value of SomeText. Use the following keywords to specify the email subject, body, header, or sender's email address as the input value:

- %SUBJECT%
- %BODY%
- %HEADER%
- %SENDER%

For example, %SUBJECT% uses the email subject as the input value. Email properties are useful when process input variables are simple data types, such as `string` or `int`.

**document, list of documents, and map of documents:**

Specify a pattern that is matched with the file name of an email attachment. A file attachment with a name that matches the pattern is used as the value for the input variable. For example, a value of `*.pdf` causes PDF file attachments to be used as input. Patterns are useful when multiple attachments must be mapped to different input variables. For example, a process requires a PDF file and a DDX file as input. Email messages that invoke the process require these files as attachments. The patterns *.pdf and *.ddx are used for the values of the input variables.

**NOTE:** Complex variable types and list or map values that do not contain document values do no appear in the Input area. You cannot specify values for these variable types.

**Output**

Specify values for output variables. The type of variable determines how you express the value.

**Document variables:**

Specify how to name files that the process returns. When processes provide `document` values as output, they are converted to files and attached to email messages. When the process returns a list or a map of documents, each document is attached to the email message. The messages are sent to the addresses that are specified in the Successful Job's Recipients property.

To name the attached files, you can include literal text, use the attributes of the returned `document` value, or both:

- Use literal text to use the same name for all attached files of every process instance. For example, the value `output.pdf` causes the names of all file attachments to be output.pdf.
- Use attributes of the returned `document` value to name the returned file based on the name of the file that the `document` value was created from:
  - **%F:**The name of the file (not including the file name extension)
  - **%E:** The file name extension

*Document attributes are assigned values when the document is created from a file. The file can be retrieved from a file system, a URL, or an email attachment. Attributes of the* `document` *value are based on the original file name.*

*Often, a process takes a document as input, manipulates the document, and returns it as output. For example, an email attachment is used as the input value and the pattern %F.%E is used as the output file name. The returned email attachment has the same name as the input file attachment.*

*If the attributes of an output* `document` *value do not have values, the %F and %E characters cause errors to occur on the server. If your process does not return an email when it completes, check the server log for error messages.*

- You can use a mixture of literal text and %F and %E characters.

  If the values of the Output properties result in files with the same name, the AEM Forms Server appends an index number to the file name. For example, a process returns three documents in a list. The Output property is out.pdf. The files that are attached to the returned email are out.pdf, out_1.pdf and out_2.pdf.

**Simple data types:**

Provide literal values for the input value of simple data types such as string, number-based types, xml, and date and time-based types. For example, the text `SomeText` is provided as a string value of SomeText.

**NOTE:** Complex variable types and list or map values that do not contain document values do no appear in the Output area. You cannot specify values for these variable types.

# 8.5. Watched folder start point properties

Use the Process Properties view to configure the following Watched Folder start point properties:

## General

**Name:**

The name of the start point. The name appears in the Service Management pages of administration console. Provide a name that is meaningful to both Workbench developers and service administrators.

**Description:**

(Optional) A description of the start point. Provide a description for other developers who edit the process. The text that you provide also appears as a description for the corresponding endpoint in the Service Management pages of administration console.

**Path:**

The watched folder location on the AEM Forms Server. The default path is /WatchedFolders/application name/process name. These folders are created in the root of the server machine upon appli-

cation deployment. In a clustered environment, the path must point to a shared network folder that is accessible from every computer in the cluster.

**Invoke Asynchronously:**

Determines whether the process is invoked asynchronously or synchronously:

- Select this option for long-lived processes, which must be invoked asynchronously.
- Do not select this option for short-lived processes, which must be invoked synchronously.

By default, this option is selected (asynchronous invocation).

**Domain Name:**

The User Management domain of the user that is used to invoke the process. The default value is DefaultDom, the default domain that is created for AEM Forms.

**User Name:**

The name of the AEM Forms to invoke the process. The user must be assigned the Services User role. This value is the user ID property, which is the name used to log in to AEM Forms. The default value is SuperAdmin.

## Server Configurations

Properties for specifying how files are handled.

**NOTE:** Performance increases when the size of the result, preserve, and failure folders are small.

**Include File Pattern:**

A pattern that matches the names of files that are used as input to the process. The value is case-sensitive. The default value of `*` includes all files. To specify multiple patterns, separate them with a semicolon (;). The following examples illustrate valid values:

- `*.pdf` includes all PDF files.
- `*.pdf;*.doc` includes all PDF and DOC files.
- `data` includes all files named data.
- `*.[dD][aA][Tt]` includes all files with the file name extension .dat regardless of letter case. For example, the pattern includes both of the files named file1.Dat and file2.dAt.

**Exclude File Pattern:**

A pattern that matches the names of files that the watched folder ignores. The value is case-sensitive. This property is useful when a folder that contains multiple files is copied to the watched folder and some of the files must be ignored.

To specify multiple patterns, separate them with a semicolon (;). The following examples illustrate valid values:

- `*.bak` excludes all files with the file name extension .bak.
- `*.pdf;*.doc` excludes all PDF and DOC files.

- `data` excludes all files named data.
- `*.[bB][aA][kK]` excludes all files with the file name extension .bak regardless of letter case. For example, both of the files named file1.Bak and file2.BAK are excluded.

**Result Folder:**

The folder where process output values are saved as files. This path can be an absolute path or a path that is relative to the watched folder. You can include the following characters, which are expanded at run time:

**%Y:**

The current year

**%M:**

The current month

**%D:**

The current day of the month

These characters are useful for minimizing folder size. The default value of `result/%Y/%M/%D stores` creates a new folder every day to minimize folder size. If the current date is October 23, 2010, results are saved in the *[watched folder]*/results/2010/10/23 folder.

If process results do not appear in this folder, look in the failure folder.

**Preserve Folder:**

The folder where input files are archived when processes that use them are successfully invoked. This path can be an absolute path or a path that is relative to the watched folder. You can include the following characters, which are expanded at run time:

**%Y:**

The current year

**%M:**

The current month

**%D:**

The current day of the month

These characters are useful for minimizing folder size. The default value of `preserve/%Y/%M/%D` creates a new folder every day to minimize folder size. If the current date is October 23, 2010, files are archived in the *[watched folder]*/preserve/2010/10/23 folder. To prevent archiving of input files, specify no value.

**Failure Folder:**

The folder where input files are copied when processes that use them are unsuccessfully invoked. These files are saved only when Preserve On Failure is selected.

This path must be relative to the watched folder. You can include the following characters, which are expanded at run time:

**%Y:**

The current year

**%M:**

The current month

**%D:**

The current day of the month

These characters are useful for minimizing folder size. The default value of `failure/%Y/%M/%D` creates a new folder every day to minimize folder size. If the current date is October 23, 2010, failed files are saved in the *[watched folder]*/failure/2010/10/23 folder.

**Preserve On Failure:**

Select to save input files in the failure folder when they are used as input for processes that fail to be invoked.

**Overwrite Duplicate Filenames:**

Select this option to overwrite existing files in the results and the preserve folders when new files with the same name are saved. When this option is not selected, a numeric index is added to file and folder names to distinguish them. This option is selected by default.

## Scheduling

Properties that specify how often the AEM Forms Server scans for new files in the watched folder. You can also specify the maximum number of files the server processes for each scan.

The configuration of Scheduling properties can affect the productivity of the AEM Forms Server. The values you provide depend on the expected frequency of new files. For example if you expect to receive 10 files every 30 seconds, server resources are wasted if the folder is scanned every second.

**Cron Expression:**

A cron expression that determines when the AEM Forms Server scans the watched folder for new files. The server you are using determines whether to use a cron expression. This property has no value by default.

When this setting is configured, the value for the Repeat Interval property is ignored. For information about formulating cron expressions, see http://quartz.sourceforge.net/javadoc/org/quartz/CronTrigger.html.

**Repeat Interval:**

The time interval at which the AEM Forms Server scans for new files. This value is in seconds. The default value of `5` causes the server to scan for new files every 5 seconds.

Unless the Throttle setting is enabled, specify a value greater than the time to process an average job. Otherwise, the system can become overloaded.

**Repeat Count:**

The number of times the AEM Forms Server scans the watched folder. A value of -1 causes indefinite scanning. The default value is -1.

**Batch Size:**

The maximum number of files or folders to process each time the AEM Forms Server scans the watched folder. This value can prevent overloading system resources. Using too many files in one scan can cause the server to crash. The default value is 2.

**Wait Time:**

The time, in milliseconds, to wait before the AEM Forms Server scans a folder or file after it begins to be copied to the watched folder. For example, the wait time is 36,000,000 milliseconds (one hour) and the file was created 1 minute ago. The file is picked up after 59 or more minutes have passed. The default value is `0`.

This setting is useful for large files or folders. Higher wait times ensure that the copying of files and folders is complete before they are used as a process input value. For example, a large file that requires ten minutes to copy requires a wait time of 10*60 *1000 milliseconds.

**Throttle:**

Select this option to ensure that the AEM Forms Server limits the number of files used per scan according to the value of the Batch Size property.

**Purge Duration:**

The number of days that files and folders are kept in the results folder before they are deleted. This property is useful for managing available space in the file system. A value of -1 causes files to remain in the results folder indefinitely. The default value is `-1`.

## Inputs/Outputs

Properties for specifying process input values and for storing process output values.

### Input

Specify a value for each input variable of the process. The type of variable determines how you express the value.

**Simple data types:**

Provide literal values for the input value of simple data types such as string, number-based types, xml, and date and time-based types. For example, the text `SomeText` is provided as a string value of SomeText.

**document, list of documents, and map of documents:**

Specify a pattern that is matched with the file name of a file that is copied to the watched folder. A file with a name that matches the pattern is used as the value for the input variable. For example, a value of `*.pdf` causes PDF files to be used as input. Patterns are useful when multiple attachments must be mapped to different input variables. For example, a process requires a PDF file and a DDX file as input. The *.pdf and *.ddx patterns are used for the values of the input variables.

**NOTE:** Complex variable types and list or map values that do not contain document values do no appear in the Input area. You cannot specify values for these variable types.

**Output**

Specify values for output variables. The type of variable determines how you express the value.

**Document variables:**

Specify how to name files that the process returns. When processes provide `document` values as output, they are converted to files and saved to the results folder. When the process returns a list or a map of documents, each document is converted.

To name the attached files, you can include literal text, use the attributes of the returned `document` value, or both:

- Use literal text to use the same name for all attached files of every process instance. For example, the value `output.pdf` causes the names of all file attachments to be output.pdf.
- Use attributes of the returned `document` value to name the returned file based on the name of the file that the `document` value was created from:
  - **%F:**The name of the file (not including the file name extension)
  - **%E:** The file name extension

  *Document attributes are assigned values when the document is created from a file. The file can be retrieved from a file system, a URL, or an email attachment. Attributes of the `document` value are based on the original file name.*

  *Often, a process takes a document as input, manipulates the document, and returns it as output. For example, a file that is copied to the watched folder is used as the input value. The pattern %F.%E is used as the output file name. The returned file has the same name as the input file.*

  *If the attributes of an output `document` value do not have values, the %F and %E characters cause errors to occur on the server. If your process does not return an email when it completes, check the server log for error messages.*
- You can use a mixture of literal text and %F and %E characters.

  If the values of the Output properties result in files with the same name, the AEM Forms Server appends an index number to the file name. For example, a process returns three documents in a list. The Output property is out.pdf. The files that are returned are out.pdf, out_1.pdf and out_2.pdf.

**Simple data types:**

Provide literal values for the input value of simple data types such as string, number-based types, xml, and date and time-based types. For example, the text `SomeText` is provided as a string value of SomeText.

**NOTE:** Complex variable types and list or map values that do not contain document values do no appear in the Output area. You cannot specify values for these variable types.

## 8.6. Programmatic start point properties

Use the Process Properties view to specify the type of client software that can invoke the process:

**SOAP:**

Select this option to enable web service client software to use the process WSDL to invoke the process.

**Remoting:**

Select this option to enable applications built with Flex to invoke processes.

**EJB:**

Select this option to enable Java client applications to use the AEM Forms Java API to invoke the process.

**REST:**

Select this option to enable client software to use the REST web service to invoke the process.

# 9. Process variables

*Process variables* enable processes to save and retrieve data at run time. You can create different types of variables for storing different types of data. You can also create variable collections for storing multiple instances of related, same-typed data.

Typically, you use a variable or a collection of variables when you need to make a decision based on the value that it holds or to store information that you need later in a process.

For example, the value in a form field may be important for making several routing decisions in the process. You can save the value of the form field in a variable to easily access the value in routing conditions.

You also use variables to save and retrieve data used to populate forms for use with the User service.

You can create variables, set variable values, and access variable values. You can set the value of variables in two ways:

- Manually at design time so that the variable has a default value each time the process is initiated. Only some variable types can have their values set manually.
- Automatically at run time so that the value of the variable can be changed during the course of a process.

Changes to variable values affect only the instance of the process in which the change occurs. For example, when a process is initiated and variable data changes, the changes affect only that instance of the process. The changes do not affect other instances of the process that were initiated previously or are initiated subsequently.

RELATED LINKS:

Variable types reference

## 9.1. Configuration parameters

Configuration parameters store data that is constant for all process instances. When a process is invoked and a process instance is created, the values of any configuration parameters are retrieved and used for the duration of the process instance.

The values of configuration parameters can be set using either Workbench or administration console. Setting the values using Administration Console is useful when it is not desirable to connect Workbench to the AEM Forms, such as in production environments.

For example, a process interacts with an FTP server. In the development environment, the process connects to a test FTP server. However, in production the process needs to connect to a different FTP server. The process includes a configuration parameter that stores the address of the FTP server. The address is used to configure the connection properties of the FTP service. When the process is moved to the production environment, administration console is used to set the value of the configuration parameter to the address of the FTP server that is used in production.

When the value of a configuration parameter is changed, all existing process instances and subsequently-created process instances use the new value.

Configuration parameters can be one of the following types:

- boolean
- int
- long
- short
- string

**NOTE:** Configuration parameters cannot be made searchable or visible in Workspace. (See Processdata exposed to users.)

To use administration console to set Configuration Parameter values, in administration console click Home > Services > Applications and Services > Service Management > *[service name]*, and then click the Configuration tab.

For information about setting the value of configuration parameters, see Applications and Services AdministrationHelp and Editing service configurations.

RELATED LINKS:

Process variables

Creating variables

Variable types reference

## 9.2. Creating variables

You create variables in the Variables view. When you create variables, consider the following practices:

- Use a common naming style.
- Use descriptive names, especially if the variables will be visible in the user interface. If you know that a variable will be displayed to a user, such as in Workspace, provide a Title string. If you choose to add a variable title, in Workspace, the title is displayed when displaying the variable in place of the variable name.
- Variables are case-sensitive. Ensure that you reference variables using the same case in your process.

When using variables, you should consider the following best practices:

- You can create as many variables as a process requires. However, to conserve database resources, use the minimum number of variables required, and reuse variables when possible.
- If you are creating an input variable and the process is to be invoked using a watched folder, the variable must be of type `document`, or a `list` or `map` variable that contains document values. (See document.)

**NOTE:** Some variable types have properties that you can configure. For example, you can configure the default values of many variable types.

## To create a variable:

1) In the Variables view, click Create New Variable .

   *You can also right-click the Variables view and select Create New Variable.*

2) In the Name box, type a name for the variable, following these naming rules:
   - Must be a valid XML element name that contains only valid XML characters.
   - Must not start with a digit.
   - Must be less than 100 characters long.
   - Must be unique and therefore cannot be `id`, `create_time`, `update_time`, `creator_id`, or `status`, which are reserved variables always in the process data model.

3) In the Title box, type a label for the variable that can be displayed in user interface windows or dialog boxes. Otherwise, the variable name is used in these locations.

4) In the Description box, type a description of the variable so that future developers can understand the purpose of the variable.

5) In the Type list, select the data type of the variable. The data type you choose depends on the values that the variable will hold and the requirements of the process. (See Variabletypes reference.)

   *If the data type is not in the Type list, click the ellipsis button, select Find Type, type a search string in the Search box, click Search, select a data type from the displayed list, and then click OK.*

   *The search string is a series of characters. Any data type that contains that series of characters is found. The search does not use wildcard characters, such as an asterisk (*).*

6) If the Type is a `map` or a `list`, in the Sub-type list, select a data sub-type. For other data types, this list is not used.

7) If the variable is used to store process data, select Process Variable in the Purpose area.
   - If the variable stores input data that is provided when the process is initiated, select Input.
   - If the variable stores data that is returned to the process initiator when the process is complete, select Output.
   - If the variable stores input data that is mandatory to initiate the process, select Required.

8) If the variable is used to store a constant data value that is used for all process instances, select Configuration Parameter.

9) To specify a length for the datatype definition, do one of the following:
   - To indicate a maximum length, select Maximum Length, and enter a numeric value in the box.
   - To indicate no limit, select Unlimited.

10) If the variable requires specific settings, type the required information in the Datatype Specific Settings area. This step only applies to some kinds of variable types. (See Variabletypes reference.)

    *If you provide a default value for a Configuration Parameter variable, when you save the process the value can be updated on the server.*

11) Click OK.

RELATED LINKS:

Editing variables

Reuse of variables

Deleting variables

Process variables

## 9.3. Creating Common Variables

Common variables are designed to enhance searchability and process tracking in Workspace. They are a set of definable variables that can be used across all applications for long-lived processes in a AEM Forms environment. Common variables are similar to process variables except that they are not local to a process and hence, its container application.

If you choose to use a use a common variable within a process, the variable will be displayed in Workspace along with variables specific to the process.

For example, if Customer ID was defined as a common variable, this would enable Workspace users to search on tasks that are all related to the same customer, spanning all processes using the Customer ID. Common variables will also appear in Workspace as column headings. You can then filter To- Do and Tracking lists based on common variable values.

1) In Workbench, click **File** > **Get Application** to access the Get Applications panel.
2) Select **Process Manager (common-variables)**and click Ok.
3) In the Applications view, check out **Process Manager (common-variables)/1.0** application and open the Process Manager (common-variables) process.

   **NOTE:** This process should only be used to define your variables. Do not add any activities to this process.
4) Create variables with meaningful names and suitable variable types. However, it is recommended that you use simple variable types. For more information, see Creatingvariables for information.
5) Check in and deploy the application to complete the creation of common variables.
6) Any variables you created will now be visible in common data tree in the XPath expression builder.

## 9.4. Editing variables

You can change the properties of variables in the Variables view.

**NOTE:** If you change a configuration variable, you can specify whether to update the variable properties on the AEM forms when you save the process.

For information about setting the value that is stored on the server in configuration variables, see Editing service configurations.

### To edit a variable:

1) In the Variables view, select a variable and click Edit Selected Variable .
   *You can also right-click the Variables view and select Edit Variable.*
2) In the Variable dialog box, edit the properties of the variable and click OK.

RELATED LINKS:

Process variables

Variable types reference

## 9.5. Deleting variables

You can delete variables that are no longer required.

### To delete a process variable:

1) In the Variables view, select a variable and click Delete Selected Variable .
   *You can also right-click the Variables view and select Delete Variable.*

RELATED LINKS:

Process variables

## 9.6. Process data exposed to users

Data stored in process variables can be exposed to Workspace users in the following ways:

- Variable data can be listed as task metadata in Workspace.
- Process variables and common variables that are simple data types can be exposed as search criteria in Workspace.
- Process variables and common variables can be exposed as column preferences in To Do and Tracking.

The values of the Enduser UI Items group of variable properties determines how variables are exposed in Workspace. (See EnduserUI items.)

# 10. Working with operations

You add *operations* to process diagrams to represent the activities that are performed in processes. Operations appear as rounded rectangles with an icon that identifies the operation.

From the services that are available, many different operations can be used to implement the process. The operation determines the activity that the AEM Forms Server performs when the operation is executed at run time. For example, when a process diagram includes an Assign Task operation, when the process is invoked and the operation is executed, the User service assigns a task to a user or group.

Most operations have properties that you need to configure so that they function properly. For example, an operation that acts on process data requires you to specify the data to use. All operations have general properties that you can configure, such as the name and description of the operation and the order in which the routes that follow the operation are evaluated. They also have properties that are specific to the operation.

RELATED LINKS:

Services

## 10.1. Adding and deleting operations

Add operations to the process diagram to define the functionality for a step in the business process. You can provide a meaningful name and description for each operation that you add to distinguish it from others. The operation name appears in the forms workflow pages of administration console.

For information about configuring operation properties, see Inputand output data for operations.

You can also delete operations that are no longer required in your process diagram.

**NOTE:** When you delete an operation, all of the associated routes and event catches are also deleted.

### Add an operation:

1) Drag the activity picker from the activity toolbar to a swimlane on the process diagram.
2) The Define Activity dialog box opens.
3) (Optional) To search for an operation, in the Find box type all or part of the operation name.
4) Double-click the service operation to add.
5) (Optional) In the Process Properties view, click the General property group and provide a name and description.
6) (Optional) Specify values for the properties in other property groups.

### Delete an operation:

1) In the process diagram, perform one of the following tasks to delete an operation.
   – Right-click the operation and select Delete Activity.
   – Select the route and in the toolbar, click Delete ✖, press the Delete key, or select Edit > Delete.

RELATED LINKS:

Providing task instructions

Working with operations

Adding elements to gateway branches

Copying elements

## 10.2. Input and output data for operations

Most operations in a process take input data, process the data or perform some action based on the data, and then return data as output. You configure operation properties to specify the input data and the location to store the output data. Typically, the properties are organized in Input and Output property groups in the Process Properties view. For some operations, other property groups are used.

The operation determines the type of data that you need to use for a property value. The operation can also allow you to specify the value in different formats.

| Format | Description | Suggested use |
|---|---|---|
| Literal value | The property value is specified by typing the value in a text box, or using a drop-down list on the Process Properties view. | The value is used for only one operation property in the process.<br>If values are used several times, you can use variables or XPath expressions to save time re-entering the same value. |
| Template | The data is specified as a string with embedded XPath expressions, where the expressions are wrapped in `{$...$}` symbols. (See Specifying templateexpressions.) | The operation property requires a string value, and the value contains both static text, and text that changes depending on the process instance.<br>For example, this format can be used when providing task instructions. (See Providingtask instructions.) |
| Variable | The data is specified as a variable that you select from a list. The value that the variable stores is used as the property value. (SeeProcessvariables | The value is already stored in a variable:<br>• The value is passed into the process when the process is invoked.<br>*You create the variable and set the value explicitly.* |

| Format | Description | Suggested use |
|---|---|---|
| XPath expression | The data is specified by an XPath expression, which you can construct using the XPath Builder dialog box. (See CreatingXPath expressions.) | The value is stored in the process data model and you want to specify either the entire variable value, or a subset of the variable data.<br>For example, you use XPath Expressions to specify the value of a specific item of data from a submitted form. (See Retrievingform field values.) |

Typically, specifying the literal value of a property is easier to accomplish than using a variable. However, specifying a variable as the property value is recommended in the following situations:

- The value is passed into the process as a process invocation parameter and stored in a variable.
- An operation returns the value and it is stored in a variable.
- You use the same value for the property of several operations in the process.

## Configuring input and output data

You configure the input and output data for an operation in the Process Properties view. The procedure in this section is general. The values you provide depend on the specific operation.

**NOTE:** Values marked with an asterisk (*) in the Process Properties view are mandatory.

**Configure the input and output data for an operation:**

1) Select the operation on the process diagram.
2) Open the Process Properties view.
3) If it exists, expand the Input category, then specify the data types and enter the input data values and properties.
4) If it exists, expand the Output category, then specify the data types and enter the output data values and properties.
5) If other categories contain input and output data, expand those categories and complete them similarly.

## Data coercion

If an operation requires data of a certain data type, and data of another data type is provided, automatic data coercion may handle the data type mismatch. When AEM Forms detects a data type mismatch, it will first attempt to resolve it by using context specific coercions implemented by custom DSCs. If this fails, AEM Forms will consult an internal coercion rules table to resolve it. Only when this is unsuccessful, a coercion exception is thrown.

For example, if an operation requires an input value of type `int` but a `string` is provided, the AEM Forms Server tries to coerce, or convert, the `string` into an `int`. If the `string` contains a numeric

value, such as the string `"123"`, the coercion works and the operation can proceed, treating the value as the integer `123`.

**NOTE:** If the current data type matches the required data type, the coercion always succeeds.

The following table represents an internal coercion rules table. It describes the definite coercion results for the specific data type pairs.

**NOTE:** In this table numeric refers to the following data types: int, short, long, byte, and float.

| Current data type | Required data type | Coercion |
|---|---|---|
| any type | string | succeeds |
| any type | binary (byte[]) | succeeds |
| any type | numeric | fails |
| any type | org.w3c.dom.Document | fails |
| any type | com.adobe.idp.Document | fails |
| any type | java.util.List | fails |
| any type | java.util.Map | fails |
| binary (byte[]) | com.adobe.idp.Document | succeeds |
| binary (byte[])<br>where byte[] is a list | java.util.List | succeeds |
| binary (byte[])<br>where byte[] is a map | java.util.Map | succeeds |
| Boolean | int | succeeds |
| Boolean | short | succeeds |
| Boolean | long | succeeds |
| Boolean | byte | succeeds |
| Boolean | float | succeeds |
| numeric | Boolean | succeeds |
| numeric | numeric | succeeds |
| document | org.w3c.dom.Document | succeeds |
| java.io.InputStream | java.util.List | succeeds |
| java.io.InputStream | java.util.Map | succeeds |
| org.w3c.dom.Document | com.adobe.idp.Document | succeeds |
| string | numeric | succeeds |

| Current data type | Required data type | Coercion |
|---|---|---|
| string | Boolean | succeeds |
| string | org.w3c.dom.Document | succeeds |
| string | com.adobe.idp.Document | succeeds |

In all the other cases, coercion will be attempted, but the result will not be known until run time, when AEM Forms will attempt to discover a context specific coercion and fail if not successful.

**NOTE:** The data coercions described in the above table are not supported by the Service's custom editors.

### Specifying template expressions

*Template expressions* are strings with embedded XPath expressions, where the expressions are wrapped in `{$...$}` symbols. This format is used when providing input or output data for operations where customized instructions are required, such as in an email message or other notification message.

*Build a template expression:*

1) In the provided space, type the string.
2) Click Insert Expression and, in the XPath Builder, select the XPath variable.
3) Repeat the steps as required.

For example, suppose you want an output data variable to contain a message to a user and you want to substitute data values where process variables are specified. The following expression could be used:

```
Dear {$/process_data/@title$} {$/process_data/@lastname$},
Your order {$/process_data/@order_id$} has been received.
```

When output, the template expression would result in the following text:

```
Dear Mrs. Jones,
Your order 12345 has been received.
```

RELATED LINKS:

Service reference

Converting data types

Creating XPath expressions

Process input and output data

Input and output data for operations

XPath Builder (Process Properties)

## 10.3. Invoking subprocesses

A service is created for each process that is deployed, Add the invoke operation of a process service to your process diagram to invoke the process. The input and output variables that are defined in the process determine the input and output properties that must be configured for the invoke operation.

You can specify how your process proceeds when it invokes the subprocess:

- Wait for the subprocess to complete before continuing.
- Continue with the next step in the process immediately after the subprocess is invoked.

**NOTE:** A subprocess can be located in the same application as the main process or in another application that is either local or remote.

### Add a subprocess:

1) Drag the subprocess from the activity toolbar to a swimlane on the process diagram.
   *The Define Subprocess Activity dialog box opens.*
2) (Optional) To search for a subprocess, in the Find box type all or part of the name.
3) Double-click the process to add.
4) (Optional) In the Process Properties view, click the General property group and provide a name and description.
5) (Optional) Specify values for the properties in other property groups. (See Processservices.)

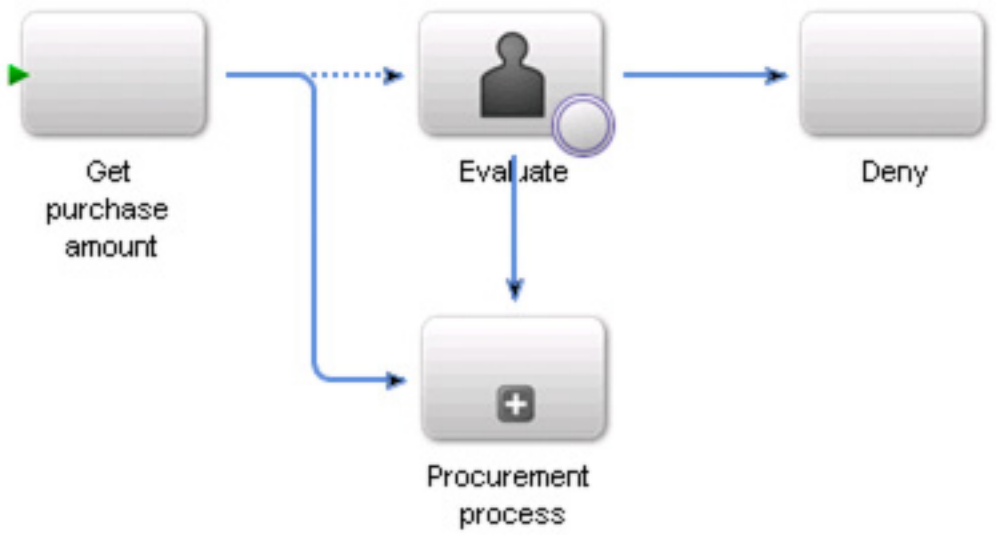
## 10.4. Abstract activities

Abstract activities are used as placeholders for operations and events in the process diagram. You use abstract activities to represent steps in the process in the early stages of process modelling:

- When you have not yet decided which operation to use in the process
- When the service that provides the operation is not yet deployed

Later in the development cycle, you define the abstract activities, which means to specify the actual service operation to use for the step. Defining an abstract activity does not disrupt any other part of the process diagram.

| Abstract element | Usage |
|---|---|
| Activity | Represents any activity that is performed in the process.<br>You define Activity elements with any operation that an available service provides.<br>If you are representing activities that are performed by a user or another process, you can use the User and Subprocess elements. The icons on these elements indicate their intended purpose. (See the information for the Subprocess and User elements in this table.) |
| Event | You define Event elements with any event type that an available service provides. |




Abstract elements are for use at design time only and provide no run-time functionality.

**TIP:** Similar to abstract activities, use abstract events to represent an undefined event type that is thrown, received, caught, or used as a start point.

## Adding, defining, and deleting abstract activities

Add an abstract activity to represent a step or event in the process. You can specify a name and description for abstract activities to indicate the functionality that is required. At a later time, you define the operation or event that provides the functionality.

You can change the operation that is defined for an abstract activity at any time. However, if you change which operation is defined, any property values that you have specified are removed. You can also delete abstract activities that you no longer require.

Abstract activities have no behavior at run time. You can test the execution of other operations in the process diagram while the abstract activities remain undefined.

Abstract activities are available from the process diagram palette.

**Add an abstract activity:**

1) Drag the abstract activity to the process diagram.

**Define an abstract activity:**

1) Right-click the abstract activity and click Define Activity.
2) In the dialog box that appears, select Select A Service Operation.
3) Double-click the service operation from the list, and then click Yes in the confirmation dialog box.

**Delete an abstract activity:**

1) In the process diagram select the abstract activity and perform one of the following actions:
   – In the toolbar, click Delete
   – Press the Delete key
   – Click Edit > Delete.

   **TIP:** You can also right-click the abstract activity and click Delete Activity.

RELATED LINKS:

Adding and defining abstract events

Copying elements

Working with operations

Drawing routes to link operations

Process diagrams

Process design guidelines

Working with operations

# 10.5. Organizing operations in swimlanes

Swimlanes and pools are used to organize process diagrams and their contents:

**Swimlanes:**

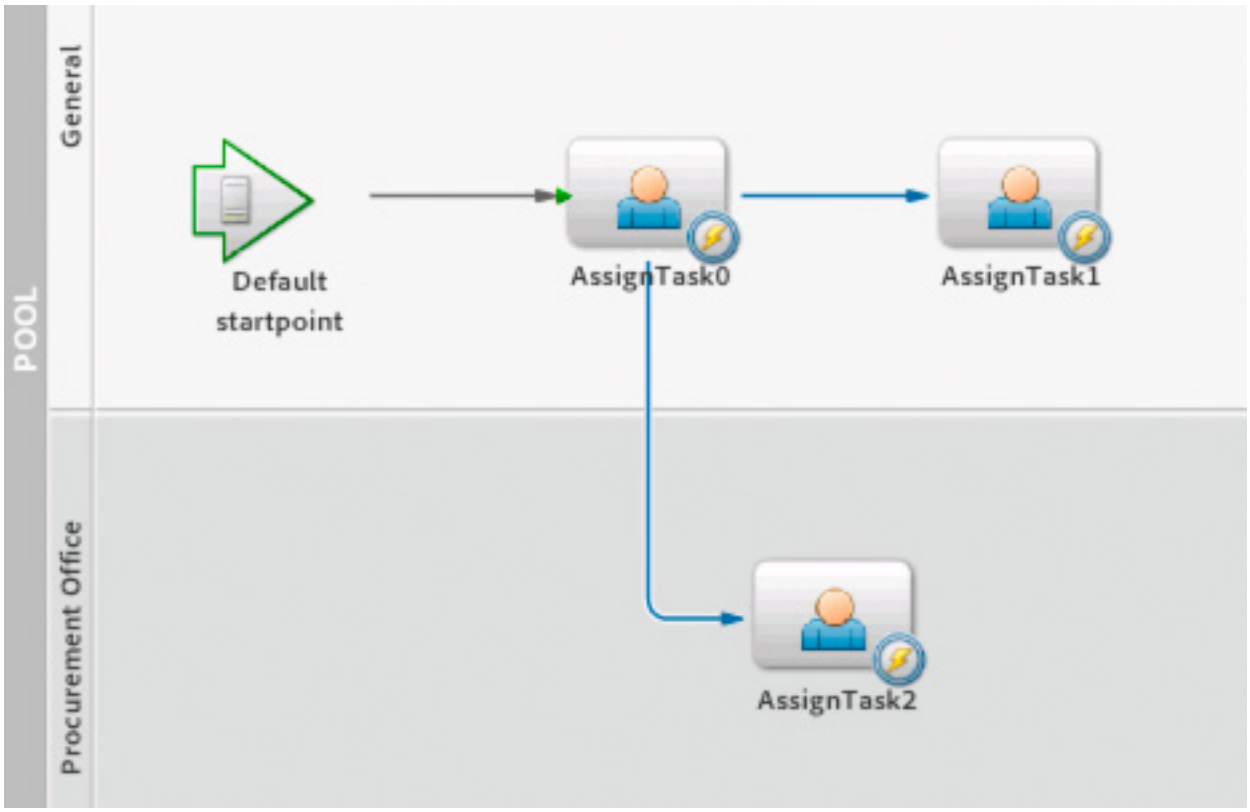
Divide the process diagram into horizontal areas that you can use to organize the items on the process diagram. Swimlanes are used to improve the readability of process diagrams and have no effect on their execution.

**Pools:**

The collection of swimlanes of a process diagram. The process editor in which process diagrams are displayed contain the entire pool.

Use swimlanes to organize your process diagrams in any way to best satisfy your needs. For example, you may want to use swimlanes to organize the operations in the process by business unit. Each swimlane represents a different business unit, and the operations are placed in the swimlane according to the business unit that completes the operation.

The following illustration includes operations in the General swimlane if a user from any business unit can perform the operation. The Procurement Office swimlane includes an operation that invokes a subprocess that is performed by that business unit.




## Adding, modifying, and deleting swimlanes

In your pool, you can add swimlanes to separate elements associated with a specific company function or role. You can modify the name, description, or color to distinguish one swimlane from another. You can also increase the height of a swimlane to improve the layout or decrease it if the default size is larger than you need.

You can also delete swimlanes that you no longer need in your process diagram. When you delete a swimlane, all elements in the swimlane are also deleted.

**NOTE:** Each pool must have a least one swimlane.

**Add a swimlane above or below the selected swimlane:**

1) In the process diagram, right-click the name of the swimlane, and then select one of these options:
   - Insert a Swimlane > Above Selected Swimlane
   - Insert a Swimlane > Below Selected Swimlane

**Modify a swimlane:**

1) In the process diagram, click the name of the swimlane to modify.
2) In the Process Properties view, click the General category and make any of these modifications:
   – To modify the name or description of the swimlane, type a new name in the Name box and a new or changed description in the Description box.
   – To modify the height of the swimlane, type a numeric value in the Height box.

   **NOTE:** You can also drag the edge of a swimlane up or down to resize it.
   – To modify the color of the swimlane, click the ellipsis button beside the Color box, click a color swatch, and click OK.

     **NOTE:** The color you choose may not be displayed in the process diagram exactly as it appears in the color swatch. The color displayed is automatically adjusted to maximize the readability of the process diagram.

**Delete a swimlane:**

1) In the process diagram, perform one of the following tasks to delete the swimlane:
   – Right-click the name of the swimlane and select Delete.
   – Select the swimlane and in the toolbar, click Delete, press the Delete key, or select Edit > Delete.

### Modifying pools

You can modify the name and description of a pool. You can give the pool a distinctive name and provide a description to better describe the purpose of your process. The name you provide for the pool is displayed in the process diagram.

**Modify the name and description of the pool:**

1) In the process diagram, click the pool.
2) In the Process Properties view, click the General category and provide a name and description:
   – In the Name box, type a name to replace the default name POOL.
   – In the Description box, type a description for the pool.

RELATED LINKS:

Process diagrams

## 10.6. Configuring the order of execution

After you add items to the process diagram, you need to specify in what order they execute by drawing routes to link the operations. (See Drawing routes to link operations.) It is a requirement that each process has an operation set as the start activity. (See Specifying the start activity of a process.)

## Specifying the start activity of a process

A *start activity* is the entry point into a process and is the first activity to occur when a process starts. Every process must have a start activity. Otherwise, the process will start and end at the same time. A start activity can be an operation or an event. There can be only one start activity in a process diagram, where the start activity appears with a green arrow on its left side.

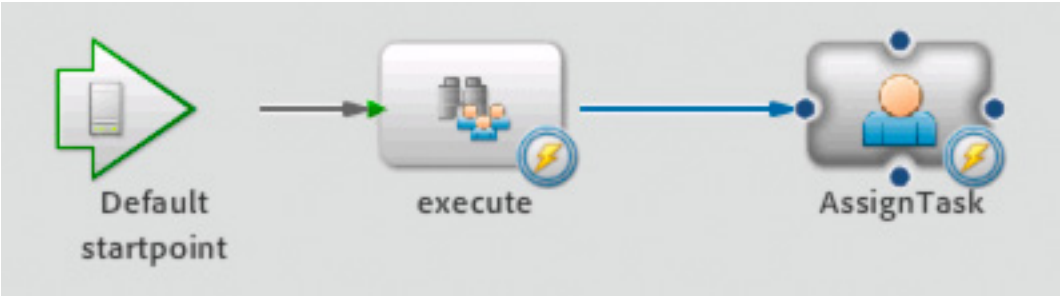
By default, the first operation you drag onto a process diagram is set as the start activity. You can then set any other operation or event in the process diagram to be the start activity. When you set the start activity, a route is automatically drawn to it from the start points.

**To specify the start activity in a process diagram:**

1) Perform one of the following actions:
   – Right-click an operation and click Set Start Activity.
   – Right-click an event and click Set Start Point.

## Drawing routes to link operations

*Routes* represent the order in which operations on the process diagram are executed. Two operations can be connected using a route to represent a sequential flow. Routes begin at one operation and end at the operation that is to be executed next. Arrowheads indicate the order of progression.




Routes can also begin and end at activity elements, events, and gateways.

Multiple routes can originate at a single operation. However, within in a branch, only one route can be followed, and a decision needs to be made to determine how to proceed:

- Each route is evaluated to determine whether it is valid.
- The first route that is found to be valid is followed.
- The order in which the routes are evaluated is configured at design time.

**TIP:** If you need operations to execute in multiple branches simultaneously, use a gateway. (See Adding-branches using gateways.*)*

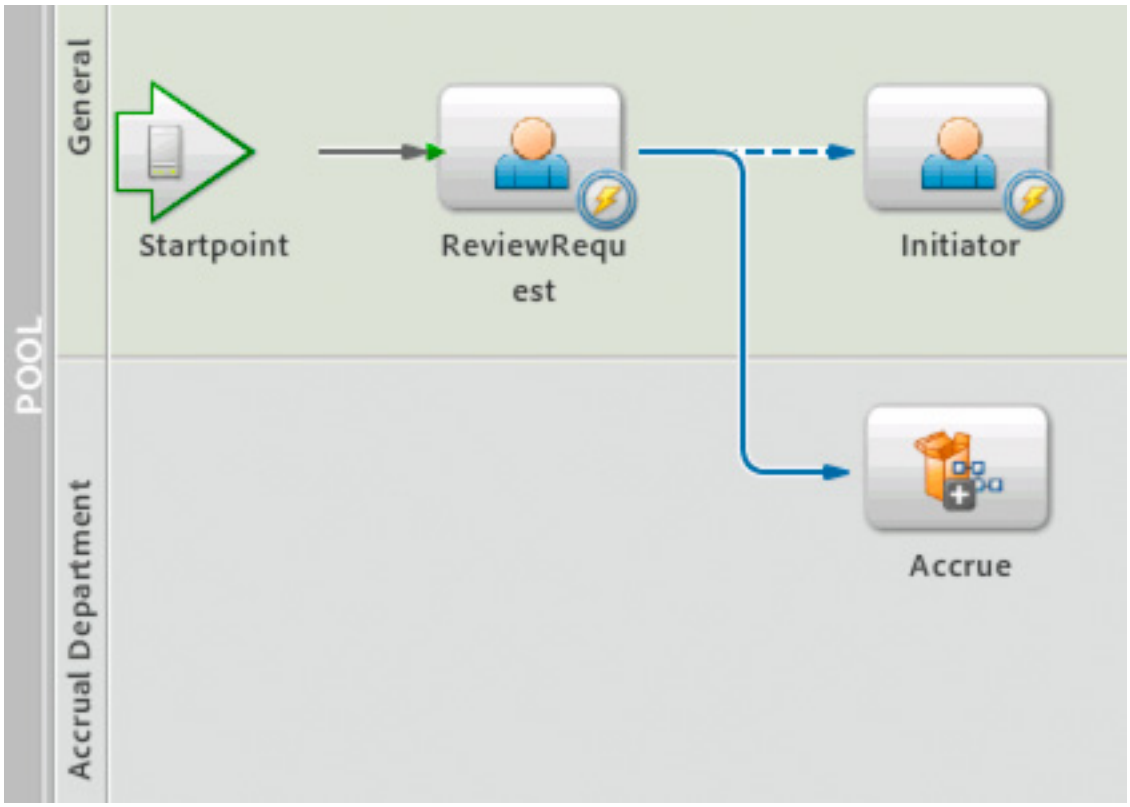
*Conditions* are logical expressions that can be associated to routes and are evaluated at run time to determine whether a route is valid. Conditions typically compare values against data that is gathered at run time so that decisions are made based on the context of the process instance. Routes can have multiple conditions. Routes that have conditions appear as dashed lines, and routes that do not have conditions are solid lines.

**NOTE:** Routes that do not have conditions are always valid.

For example, a company's internal process for requesting purchases is automated using AEM Forms. To start the process, a user fills and submits a form using Workspace. The user's manager opens the form in Workspace and selects an option to either decline or approve the purchase. The decision determines the next operation in the process, which is to either return the form to the originator (the purchase is declined) or to send the form to the accrual department (the purchase is approved).

On the process diagram, the decision is represented by two routes that originate at the same operation. One route has a condition associated with it that evaluates whether the manager denied the purchase. If the condition is true (the request was denied), the route is followed. If the condition is false, the other route is evaluated.

Dashed route indicates a condition




When no valid routes are found after an operation is complete, the process is complete. Therefore, when multiple routes originate at an operation, typically one of the routes is not given a condition and is evaluated last. This route is followed by default if the other routes are evaluated and found to be invalid. This design ensures that the process continues to progress after the operation is complete.

Each route has a name that you can customize to provide meaning. Typically, the name provides insight into the condition that is associated with the route. Route names can also be exposed as form submission options for Workspace users. (See Providingactions for submitting tasks.)

### Adding and deleting routes

You add routes between elements in a process diagram to specify the order in which they are executed.

You can draw one or more routes from any element. You can also draw routes to the same originating operation, event gateway element, or activity element to represent iteration. To draw a route between two elements, both elements must already exist in the process diagram.

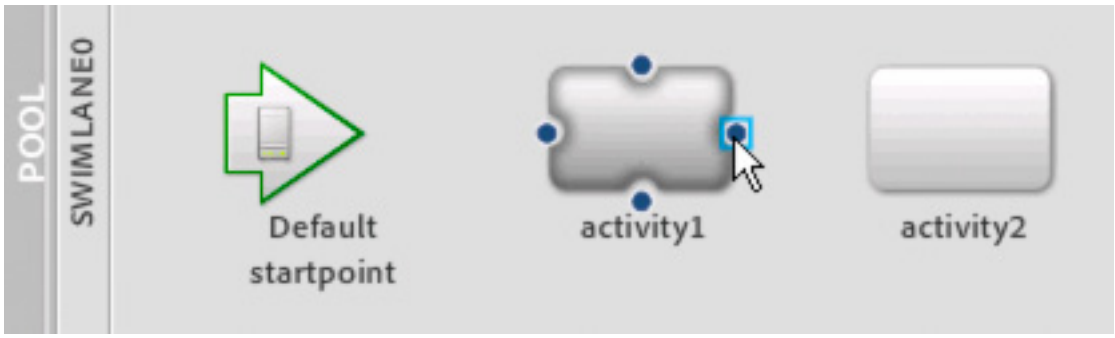
Default route names do not appear in the process diagram. You must give the route a name for text to appear for a route. (See Modifying route labels.)

**NOTE:** When you add an event as a start point, a route is drawn to the element that is set as the Start Activity.

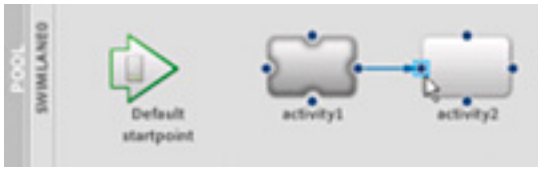
You can delete any routes that you longer no require in your process.

*To add a route:*

1) In the process diagram, select an existing operation, event, activity element, or gateway element where you want the route to begin. The route anchors appear on the element.
2) Pause the pointer over a route anchor until the pointer changes to a pointing hand icon.




3) The anchor that you use to begin drawing the route determines the direction of the route and the order in which the route is executed.
4) Drag the pointer to the operation, event, activity element, or gateway element until you see the hand icon appear on the anchor where you want to terminate the route.




*A route appears that connects the two elements.*

5) (Optional) In the Process Properties view, provide a name and description:
   – In the Name box, type a name for the route.
   – In the Description box, type text to describe the route.

*To delete a route:*

1) In the process diagram, delete the route by performing one of the following tasks:
   – Right-click the route and select Delete Route.
   – Select the route and in the toolbar, click Delete , press the Delete key, or select Edit > Delete.

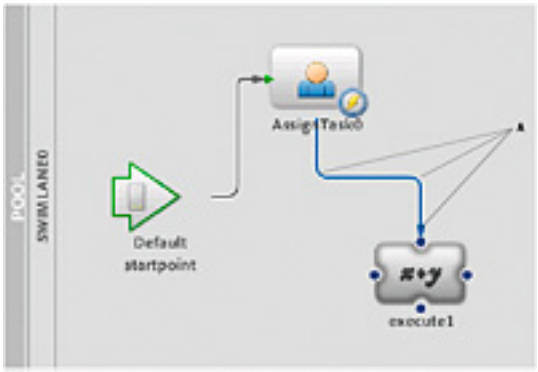
### Modifying route shapes

You can modify the shape of a route by dragging the originating element or terminating element from one location to another in your process diagram.

This action will modify the shape of your route. For example, a route drawn from one operation to another in a process diagram may look like the following illustration.

After moving the execute1 operation to the right, the route's shape changes to accommodate the new position of the operations.

Grip points




You can also select the route and drag any grip point to modify the shape of the route. A *grip point* is a black dot that appears on the selected route.

If you need to reorder your operations, you can drag the start or the end point of the route to another element in the process diagram. To do that, pause your pointer over the grip point at the beginning or end of the route until the hand icon appears, and then drag the point to another element.

### Modifying route labels

You can modify the text and position of route labels to make the process diagram more meaningful. When you add a route, default text is provided. A route label only appears when you modify its default name.

**IMPORTANT:** Any route label name that has the text "route" will not appear in the process diagram.

*To modify the text and position of a route label:*

1) In the process diagram, select a route and modify its label text or position, or both:
   – In the Properties box, in the Name box, type a new name for the route.
   – Move the route label from its default position on the route to another area in the process diagram.

RELATED LINKS:

Adding and deleting operations

Configure event start points

Adding branches using gateways

Drawing routes to link operations

Copying elements

Routing condition format

Creating XPath expressions

Process execution

## Making decisions using routes

This section provides information about the following topics:

RELATED LINKS:

Routing condition format

Adding and modifying routing conditions

Specifying the order of routes

### Routing condition format

Routing conditions consist of three parts, in the following format:

```
expression1 operator expression2
```

Expressions consist of values from the process schema or form data, or are derived from data values by using functions. The operator defines the required relationship between the two expressions. The expressions must be of the same data type.

For example, if you want to follow a route only if the value in a purchase amount form field is greater than 5000, the condition would be as shown here:

```
PurchaseAmt > 5000
```

You can create complex expressions for specific routing conditions to adhere to the required business logic in your processes.

Routing conditions cannot be set on routes that originate from a start point event.

### Adding and modifying routing conditions

To add a condition, a route must already exist in the process design. You can modify routing conditions any time after you create them. You add and modify routing conditions in the Process Properties view.

*To add or modify a routing condition:*

1) In the Process editor, select a route.
2) In the Process Properties view, click the Conditions category.

3) If you are adding a routing condition, click Add Route Condition . If you are modifying a routing condition, select an existing condition and click Edit Selected Route Condition . The Route Properties dialog box appears.
4) In the Expression box on the left, type the first part of the expression. If the condition is complex, click the ellipsis button to display the XPath Builder.
5) In the Operation list, select an operation.
6) In the Expression box on the right, type the second part of the expression. If the condition is complex, click the ellipsis button to display the XPath Builder.
7) Click OK to close the Route Properties dialog box.
8) If you have more than one routing condition in the Conditions category, select the join condition to determine how the conditions are evaluated:
   – Use AND Join For Conditions means the route is valid only if all the conditions evaluate to `True`.
   – Use OR Join For Conditions means the route is valid when one or more of the conditions evaluate to `True`.

   *The default join condition is Use OR Join For Conditions.*

### Specifying the order of routes

When several routes originate at one step in a process, you need to specify the order in which the routes are evaluated. The process follows the first route where the routing condition evaluates to `True`.

Specifying the order of the routes is important when multiple routes can potentially evaluate to `True`, but one route should be taken instead of the others according to the business logic you want to implement.

You specify the order of routes in the Process Properties view.

*To specify the order of routes:*

1) Select an operation or event from which multiple routes originate.
2) In the Process Properties view, click the Route Evaluation category. The routes are displayed in the order they are evaluated, first to last.
3) To sort the routes, select a route and click the Move Route Up or Move Route Down button.

RELATED LINKS:

Making decisions using routes

Drawing routes to link operations

Adding branches using gateways

Making decisions using routes

Creating XPath expressions

Making decisions using routes

XPath Builder (Process Properties)

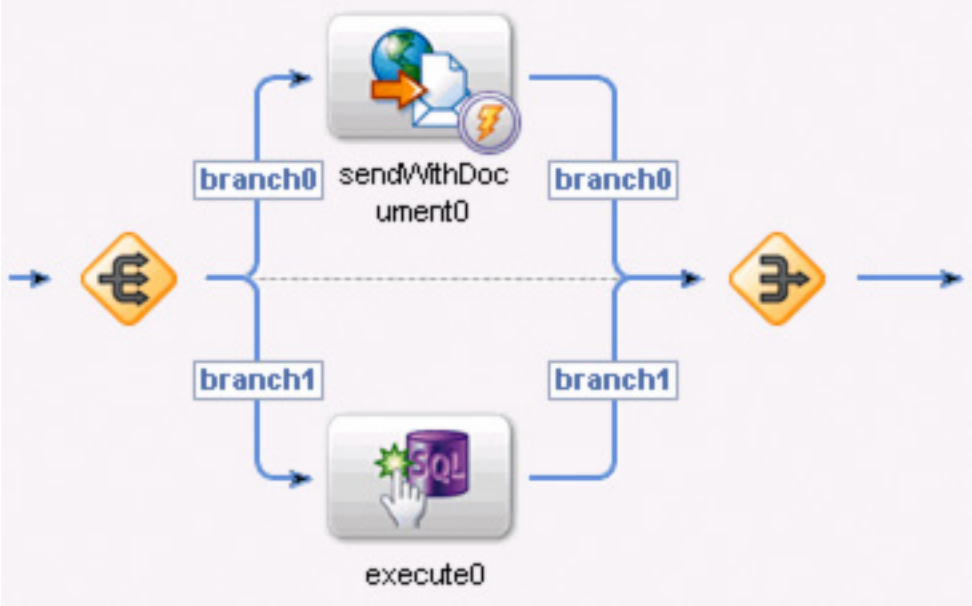
XPath function reference

# 10.7. Adding branches using gateways

*Gateways* contain one or more branches and are used to control the divergence and convergence of sequence flow. You typically add a gateway for one of the following reasons:

- You want to execute operations simultaneously.
- You want to use a branch type that is different from that of the main branch. (See Branches.)

The following diagram shows a Gateway element with two branches.




You also control how the process continues after the gateway, where the branches converge:

- The process continues only after all of the operations of each branch are complete. This configuration is used when subsequent operations in the process depend on the completion of all branches in the gateway.
- The process continues when the operations in any one branch are complete. This configuration is used when the process is affected by which branch is first completed.
- The process continues immediately, regardless of whether any of the branches are complete. This configuration is used when subsequent operations do not depend on the results of the operations in the gateway.

## Adding and deleting gateways

Add a gateway element to your process diagram to execute branches in parallel. When you add a gateway, it contains one branch and no other elements. You can add elements and additional branches as required. You can rename or add a description to the gateway to distinguish it from others in the process diagram.

**NOTE:** You cannot add gateway elements to short-lived processes. If a branch within a gateway is transactional, you cannot add a gateway to it. However, you can add a gateway to a main branch that is transactional. (See Branches.)

You also configure when the main branch proceeds after the gateway element is executed. The Control Type property controls this behavior, and can be set to the following values:

**AND-WAIT:**

The main branch proceeds after all of the branches in the gateway are complete. AND-WAIT is the default value.

**OR-WAIT:**

The main branch proceeds after only one of the branches in the gateway is complete.

**NO-WAIT:**

The main branch proceeds immediately after the branches in the gateway are started. The main branch does not wait for the branches to complete.

You can delete the gateway element from the process diagram at any time. When you delete a gateway element, all routes to and from the gateway element are deleted. All branches in the gateway are also deleted.

**To add a gateway element:**

1) Drag a Gateway from the toolbar to a location in your process diagram. If the process is short-lived, you are prompted to change it to long-lived.
2) (Optional) In the Gateway Properties dialog box, provide a name and description:
   – In the Name box, type a name for the gateway element.
   – In the Description box, type text to describe the gateway element.
3) In the Control Type list, select the control type to use.
4) Click Save.

**To delete a gateway element:**

1) In the process diagram, perform one of the following tasks to delete a gateway element:
   – Right-click the gateway element and select Delete Gateway.
   – Select the gateway element and in the toolbar, click Delete, press the Delete key, or select Edit > Delete.

RELATED LINKS:

Adding branches using gateways

Adding, editing, deleting, and copying branches

Short-lived processes and long-lived processes

Adding elements to gateway branches

## Branches

A *branch*is a set of activities that the process executes. Process diagrams can include one or more branches. When you create a process and add elements to the process diagram, the elements belong to the default branch, called the *main branch*.

The use of multiple branches allows you to implement different behaviors for different parts of the process. You add branches to a process by drawing routes from activities and other elements to gateways. (See Addingbranches using gateways.) After you add a branch to a gateway, you can configure its branch type.

The branch type also dictates the type of operations that can be added. For more information about branch types, see Transactions.

At run time, operations in a branch are executed sequentially. When an operation is running, no other operation in the branch is running. When the operation is completed, the next operation is executed. To execute multiple operations simultaneously, they need to belong to different branches inside gateways.

The type of the main branch is specified when the process is created and can be changed in the properties of the process version.

RELATED LINKS:

Operation and branch compatibility

## Adding, editing, deleting, and copying branches

You create branches that execute in parallel by adding process elements to a gateway element. You can add as many branches as your process requires. You can rename a branch and add a description to make it distinct from other branches. The branch type you configure specifies how the operations in the branch behave at runtime. The branch type is synchronous by default. (See Transactionsand branches.)

After you add a branch, you can edit that branch by adding additional elements to define your business logic. (See Addingelements to gateway branches.)

You can copy branches and add them to another branch, such as the main branch or a branch in a gateway element. You can also copy the branch to the current or a different process diagram. When you copy a branch, you copy only the process elements and routes that join them, and not the branch properties.

You can also remove branches from your process diagram.

**To add a branch:**

1) In the process diagram, select a process element to include in the branch. The route anchors appear on the element.
2) Place the pointer over a route anchor to highlight it.
3) Drag the pointer to the start or end of the gateway. If your route connects the element to the end of the gateway, a route from the element to the start of the gateway is created.
4) Click Save.

**To edit a branch:**

1) In the process diagram, click the branch label to select it, and then enter the new name for the branch.
2) In the Process Properties view, edit other branch properties as required. (See Transactionsand branches.)

**To copy a branch:**

1) In the process diagram, right-click the gateway element, select Copy Branch, and then select the name of the branch to copy.
2) In the toolbar, do one of the following actions:
   – In the toolbar, click Paste .
   – Select Edit > Paste.
   – Press Ctrl+V.
3) Move the branch elements to the appropriate location in the process diagram.
4) Create routes as required to add the copied branch to the process.

**NOTE:** When you copy a branch, the routes that join it to the gateway are not copied.

**To delete a branch:**

1) In the process diagram, right-click the gateway element, select Delete Branch, and then select the name of the branch to delete.

   **NOTE:** If you delete all the elements in a branch, the branch is deleted automatically.

RELATED LINKS:

Adding branches using gateways

Branches

Process execution

Short-lived processes and long-lived processes

## Adding elements to gateway branches

You add elements to a branch by drawing routes from an operation, event, or activity to the gateway or existing branch elements. Use the tools and methods you use for other parts of the process diagram to add, remove, and change the sequence of branch elements as required. (See Drawingroutes to link operations.) You can add any number of elements to a branch.

**TIP:** You can drag one end of the gateway element to increase the distance between the two ends. Increasing the distance creates space for multiple branches and elements.

You can modify the routes connected to the start and end of the gateway using the methods used for regular routes. For example, you can change which operation is executed first by dragging the anchor for the first route in the branch to a different operation. However, when you connect an element to the end

of the gateway, if no route connects the branch elements to the start of the gateway, one is added automatically.

If you do not provide a final route to connect the branch to the end of the gateway, one is added automatically the next time you open the process.

**To change which operation is executed first:**

1) In the branch, right-click the operation to execute first and select Set Start Activity.

### Validating routes used by gateway branches

When you add routes to gateway branches, an error icon appears when you attempt to draw invalid routes. For example, you cannot create a route that connects a branch activity to an element outside its branch. Routes that connect elements in different branches are also invalid.

Additional route validation information appears in the Validation Report view. For example, a message appears in the Validation Report view if a branch within a gateway is incomplete. (See Validationreports).

### Operation and branch compatibility

Not all operations are compatible with all branch types. Operation and branch compatibility may help you to decide the type of branch that you should use in your process.

**NOTE:** You use gateways to add additional branches to a process.

| Branch type | Compatible operations |
|---|---|
| Asynchronous | All operations |
| Synchronous | All operations |
| Transactional | Set Value, Stall, and Execute Script service operations |

**NOTE:** The operations of the User, Wait Point, and Execute Script services, as well as gateway elements, cause synchronous branches to behave as asynchronous branches.

### Transactions

A *transaction* is a context in which one or more operations are executed. Generally, the operations that are executed within a transaction are not committed until all of the operations are executed successfully.

If one operation in a transaction fails to execute successfully, the transaction is rolled back and the effects of any operations that were already executed in that transaction are reversed.

In Transactional branches, the operations in a branch are executed in a single transaction. In an Asynchronous or Synchronous branch, each operation is executed in separate transactions.

*Transactions and branches*

Transactional branches are useful when a common goal is dependent on the successful execution of a series of operations. If an operation in a transactional branch executes with errors, the branch is stalled. When the branch is retried, the transaction for the branch is rolled back and the first operation in the branch is executed again.

For information about the behavior of actions in Transactional branches, see Transactionsand operations.

Asynchronous and synchronous branches are useful when the failure of one operation in the branch does not devalue the successful execution of previous actions in the branch. If an operation in an Asynchronous or Synchronous branch executes with errors, the branch is stalled. When the branch is retried, only the failed operation is rolled back and the operation is executed again.

### Branch behavior when errors occur

The behavior of a branch when a stalled branch or a stalled operation is retried is a factor to consider when deciding which type of branch to use in your process.

For all branch types, when a branch exception occurs, the branch is stalled and, when an operation exception occurs, the operation for which the exception occurred is stalled.

The following table describes how the different branch types continue execution after the forms workflow administrator retries the stalled branch or operation.

| Branch type | Behavior when a stalled branch or operation is retried |
|---|---|
| Asynchronous | The process continues from the point in the process where the exception occurred. |
| Synchronous | The process continues from the point in the process where the exception occurred. |
| Transactional | The branch is rolled back, and the process continues from the first operation in the branch. |

*Transactions and operations*

Operations are handled differently in transactions, depending on whether the operations are transaction-aware and long-lived.

### Transaction-aware operations

Operations that are transaction-aware will participate in a transaction. When a Transactional branch is rolled back, the operations in the branch that are transaction-aware are returned to their original state before that branch was executed.

If an operation is not transaction-aware and it is used in a Transactional branch that stalls, the effects of the operation are not rolled back when the branch is retried. The operations are executed a second time. For example, if you use the Email operation in a Transactional branch, each time the branch stalls and is retried, another email is generated.

The only transaction-aware operation that Foundation provides is the execute operation of the Set Value service. However, your development team may create transaction-aware operations using the AEM Forms SDK.

### Long-lived operations

Long-lived operations execute independently of the branch to which they belong:

- When a Transactional branch is stalled and is retried, any long-lived operations in the branch continue to run and are not rolled back.
- If a computational error occurs during the execution of a long-lived operation, the operation is stalled, but the branch is not stalled even if the operation belongs to a Transactional branch.

None of the operations that Foundation provides are long-lived. However, your development team may create long-lived operations using the AEM Forms.

RELATED LINKS:

Transactions and branches

Short-lived processes and long-lived processes

SetValue

Stall

ExecuteScript

WaitPoint

Short-lived processes and long-lived processes

Transactions and branches

Transactions and operations

Adding branches using gateways

Branches

Adding, editing, deleting, and copying branches

Adding and deleting routes

Adding and modifying routing conditions

# 11. Controlling process flow using events

The AEM Forms Server provides an event framework that enables business events to be defined. *Events* provide processes with a mechanism for interacting with the event framework.

Events are represented on process diagrams as circle-shaped icons. Routes can be drawn from events to other elements on the process diagram so that the occurrence of the event alters the progression of the process.

Event in a process diagram.

You can use Asynchronous, Timer, and Exception events types. Each event type is either a service-defined event or a custom event. When adding a custom event to a process diagram, select an event that is located in the current application or in another application. The other application can be either local or remote. To use custom events from remote applications, the applications must be deployed on the AEM Forms Server.

Processes can receive notifications about events that occur during process execution. You can configure the conditions to respond to for each notification you receive. Processes can throw events to provide notifications to other processes about an occurrence in the process. As part of sending a notification, you can send specific details about the event or include additional data.

You can also use events to catch errors that occur on the server upon process execution, handle situational errors, and to insert time-based execution constraints on operations. Events can also be used to pass information from process to process using a pre-defined data schema.

RELATED LINKS:

Eventtypes

Eventcategories

## 11.1. Event types

An *event type* is the definition of a situation that can occur. A unique name identifies each event type. Event types can be active or inactive on a server and can be used only when active.You can use the following event types:

- Service-defined events: Services can provide events and appears in the Events view when the service is activated.
- Custom events: You can create these event types using Workbench. It is useful to create an event type if an existing one does not meet the requirements of your process.

When an event occurs at runtime, use event types to provide a notification about the occurrence of the event (throw the event) and to receive the notification (receive or catch the event).

Event notifications carry data with them so that information about the event is communicated by the event thrower to the event receiver. Events distinguish between two types of data:

**Event data:**

A small set of discreet data items that event receivers use for decision-making purposes.

**Event message data:**

The payload of asynchronous events is in the form of a larger amount of XML data. Typically, this data either cannot be altered during the process because it is a document of record or the process requires the XML document structure to be validated. For example, the process for a company's internal purchase order request throws an event when a purchase order requires approval. The event message data is the original purchase order form that needs to be approved.

RELATED LINKS:

Eventcategories

Displayingthe properties of event types

Creatingcustom event types

# 11.2. Event categories

*Categories* are form groups of events types that have different behaviors and are used for different purposes. There are three default categories of events:

**Asynchronous:**

Asynchronous events provide event information in messages. Processes receive messages for asynchronous events that they receive and catch, and provide messages that they throw.

**Exception:**

Exception events provide notifications about errors that occur on the AEM Forms Server. Processes can receive and catch exception events but cannot throw them.

**Timer:**

Timer events are notifications about time-based situations. When a process throws a timer event, the throw occurs after a specific amount of time has passed. Similar to asynchronous events, timer events also provide event information in messages.

You can also organize events into groups so that you can more easily manage a set of event types.

RELATED LINKS:

Eventtypes

Organizingevent types

# 11.3. Throwing events

Event types are thrown to provide notifications about the occurrence of the event. Service operations can throw exception events that indicate that an error has occurred during execution of the operation. You cannot explicitly throw a service exception or exception event types.

Many services provide asynchronous events that you throw in short-lived and long-lived processes. When you throw an event in a process, you configure the event data and the event message data that is provided in the event notification.

An event throw appears as an icon on the process diagram. Routes can begin and terminate at an event throw similar to the way they do for operations.

At run time, when the throw of an event type is routed to and executed, the notification about the event is sent to all of the receivers and catchers of that event type. After the event is thrown, the process continues according to the routes that originate from the event throw.

To configure event throws, you can specify the event data that is provided to event receivers. For asynchronous event types, you can also specify the event message data. For timer event types, you can specify the timer data.

## Add and configure asynchronous event throws

You can add an event throws to short-lived processes.

***NOTE:*** *You cannot throw exception event types.*

1) Drag the Event Picker from the Activity Toolbar to an unused part of the process diagram.
2) (Optional) In the Name box, type a new name to replace the default name.
3) (Optional) To search for a specific event, replace the default string in the Find box with a value.
4) Select an asynchronous event type and click OK.
5) In the Event Behavior Configuration dialog box, ensure that the Event Throw is selected.
6) Add event data to send. (See Configurethe event data to send.)
7) (Optional) Send additional XML data from an XML variable by completing the following steps:
   – In the Event Behavior Configuration dialog box, click the ellipsis beside the Event Message Data box.
   – In the Asynchronous Event Throw Configuration dialog box, select the XML variable that stores the data and click OK.
8) Click OK.

RELATED LINKS:

Addingevent throws

Receivingevent throws

Catchingevent throws

## Add and configure timer event throws

You can configure timer event throws to provide timer data. You can provide a specific date or duration.

1) Drag the Event Picker from the Activity Toolbar to an unused part of the process diagram.
2) (Optional) In the Define Event dialog box, in the Name box, type a new name to replace the default name.
3) (Optional) To search for a specific event, replace the default string in the Find box with a value.
4) Select an event type from the Timer event category and click OK.

   **NOTE:** The Timer category appears only after you create and deploy at least one custom timer event. AEM Forms does not provide default system timer events. Custom timer events that are not deployed are listed in the current application category. (See Creating custom event types.)
5) In the Event Behavior Configuration dialog box, ensure that the Event Throw is selected.
6) Add event data to send. (See Configurethe event data to send.)
7) Provide the date and time data by completing one of the following steps:
   - To set the duration of the timer, select Set Timer Event Duration enter the duration in the following fields.
     - **Date**: Use the time specified in the XPath expression to a `DateTime` value.
     - **Hours, Minutes, Second**: Use the defined time from the Hours, Minutes, and Seconds fields.
   - To set a specific date when the timer expires, select one of the following values from the list:
     - **LITERAL**: Click the ellipsis button beside the Date box, and in the Calendar Date Selector dialog box, choose a date. Also, enter the Hour, Minute, and Seconds values and click OK.
     - **XPATH**: Click the ellipsis button beside, in the Timer Event Throw Configuration dialog box, build an XPath expression to represent the date as a `DateTime` value.

RELATED LINKS:

Eventtypes

Eventcategories

Addingevent throws

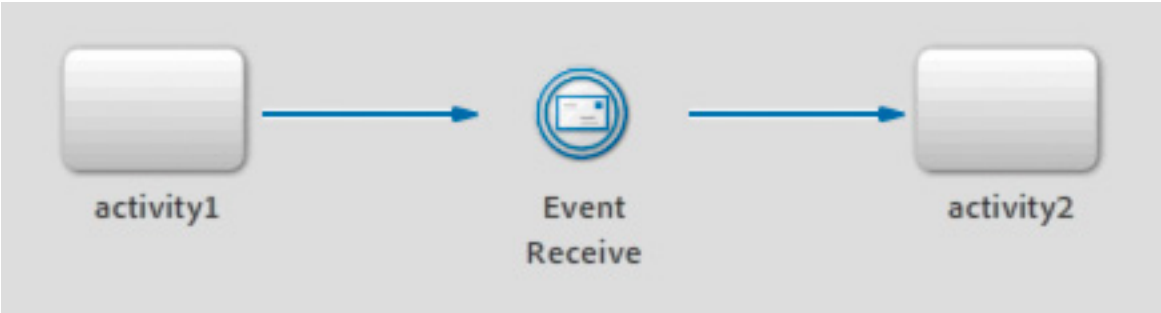
Receivingevent throws

Catchingevent throws

# 11.4. Receiving event throws

Processes receive events to react to events when they are thrown. *Event receives* are activities that you add to the process diagram of long-lived processes. Event receives react to event throws during the process that they belong to is executing. ***NOTE:*** *Event receives cannot be added to short-lived processes.*

Event receives appear as icons on the process diagram. Event receives can have routes that begin or terminate at the event in the same way that operations do. When a route is followed to the event receive, the event receive waits for an event to be thrown. The process continues only after the event is received and acted on according to the event filters that are configured on it. Event filters define the criteria to which your event receive responds. For example, you can configure your event to respond only when the event message data for field exceeds a value.




Event receives can receive event throws in another process or application, in the same process, or in an Adobe Flex application that uses the event framework.

RELATED LINKS:

Controllingprocess flow using events

Addingevent throws

Addingevent receives

## Add and configure event receives

You can add an event receives to long-lived processes for timer, asynchronous, and exception event types.

1) Drag the Event Picker from the Activity Toolbar to an unused part of the process diagram.
2) (Optional) In the Name box, type a new name to replace the default name.
3) (Optional) To search for a specific event, replace the default string in the Find box with a value.
4) Select an event type from one of the event categories and click OK.

   **NOTE:** The Timer category appears only after you create and deploy at least one custom timer event. AEM forms does not provide default system timer events. Custom timer events that are not deployed are listed in the current application category. (See Creatingcustom event types.)
5) In the Event Behavior Configuration dialog box, select Event Receive.
6) (Optional) Add event filters. (See Definingevent filters.)
7) (Optional) Store event data to process variables. (See Storingevent data.)

8) Click OK.

RELATED LINKS:

Controllingprocess flow using events

Addingevent catches

Receivingevent throws

Addingevent receives

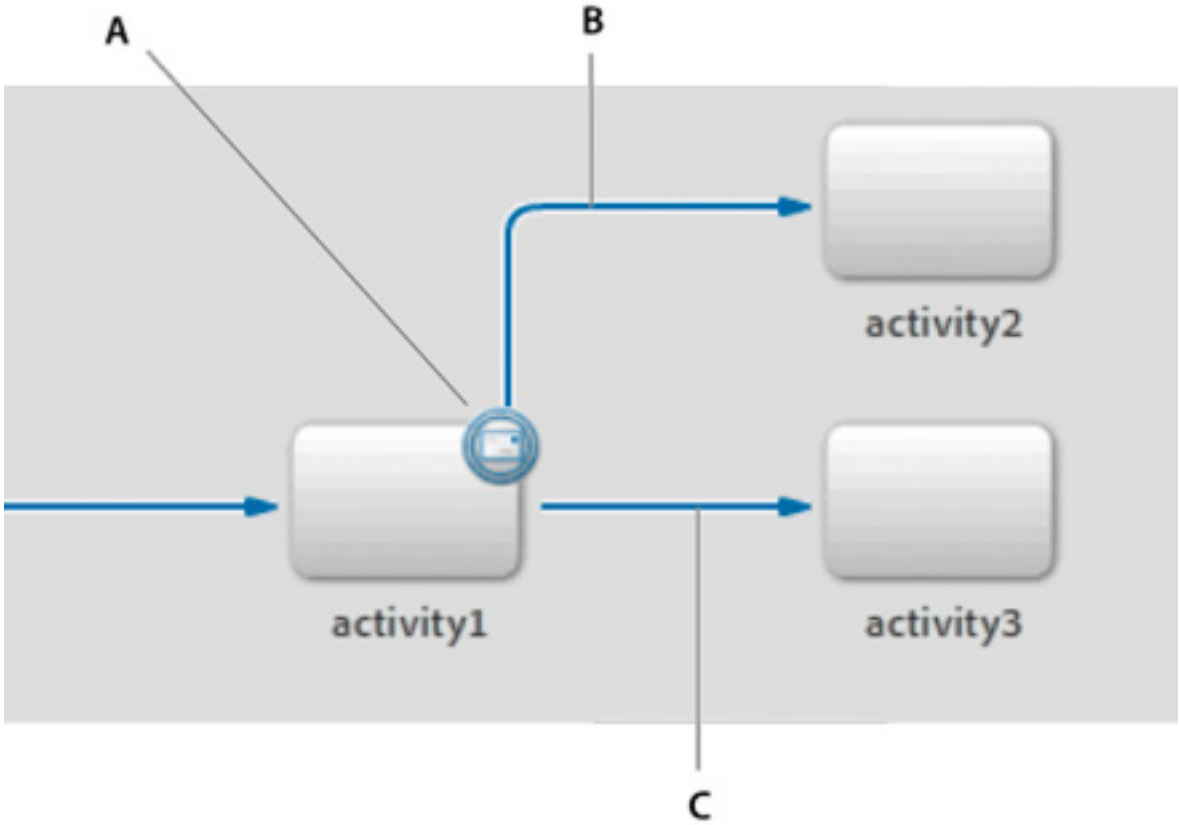
# 11.5. Catching event throws

Processes catch events to react to event throws that occur during the execution of an operation. You can configure filters for catches of asynchronous and timer events that determine whether the event is acted on. Event filters are based on the information in the event data. Furthermore, the event data and the event message data can be saved as process data. When you catch exception events, you specify the type of exception that is acted on.

Event catches are attached to operations on the process diagram and appear as icons in one of the corners of the operation icon. Operations can catch a maximum of four event types.

The *event catch* reacts to thrown events only while the operation that it is attached to is executing:

- If the event is caught during the execution of the operation, the route (if any) that begins at the event catch is followed, and the operation stops executing.
- If the operation completes and the event type that is being caught is not thrown, the usual evaluation of routes that originate at the operation occurs.




**A.**

Event catch

**B.**

Route followed if event is caught

**C.**

Route followed if operation is complete and no event is caught

Many service operations include an exception event catch by default, which is used to handle exceptions that occur when the operation executes. The developer of the service defines the exception events that are thrown. For information about the exceptions that each operation throws, see Servicereference.

## Add event catches

Add an event catch to an operation to react to the occurrence of the event while the operation executes. You can add up to four event catches to an operation, but you cannot add Exception event catches to operations.

If Operations include an Exception event catch, it is provided by default. You cannot replace exception event catches with other event catches.

After you add the event catch, you can specify which operation executes when the event occurs. (See Handlingevent catches).

1) Drag the Event Picker from the Activity Toolbar to an empty corner of an operation or Activity element.

*After the event is attached to the operation, the operation looks similar to the following illustration.*

2) (Optional) In the Name box, type a new name to replace the default name.
3) (Optional) To search for a specific event, replace the default string in the Find box with a value.
4) Select an event type from one of the event categories and click OK.

   **NOTE:** The Timer category appears only after you create and deploy at least one custom timer event. AEM forms does not provide default system timer events. Custom timer events that are not deployed are listed in the current application category. (See Creatingcustom event types.)
5) In the Event Behavior Configuration dialog box, select Event Receive.
6) (Optional) Add event filters. (See Definingevent filters.)
7) (Optional) Store event data to process variables. (See Storingevent data.)
8) Click OK.

RELATED LINKS:

Catchingevent throws

Controllingprocess flow using events

Catchingevent throws

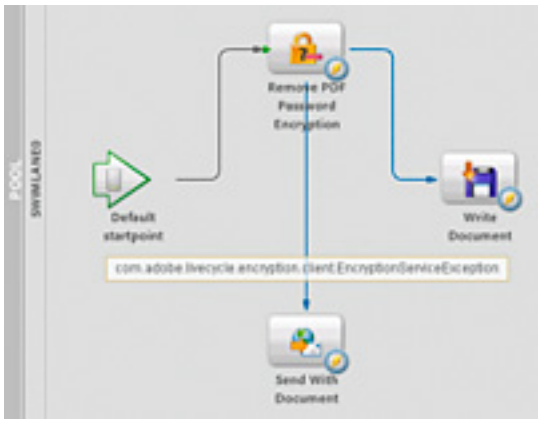
Addingevent catches

Controllingprocess flow using events

### Handling event catches

To handle event catches, you draw one or more routes from the event catch to another step in the process diagram.

For example, to handle a situation where an operation fails and normal processing cannot continue, you can draw a route from the operation's exception event catch to a Send With Document operation that the Email service provides. The operation sends an informative email to a system administrator to investigate the problem.




When you draw a route from an exception event catch, you specify the exception that the event catch reacts to.

#### Handle event catches

1) In your process diagram, add the step or steps that are to handle the event catch.
2) Select the operation that has the event catch and draw a route from the event catch symbol to the operation that handles the event catch.
3) (For Exception event catches only) In the list of exceptions in the Select Service Operation Fault dialog box, select an exception and then click OK.
4) Repeat these steps to add multiple routes from the event catch.

RELATED LINKS:

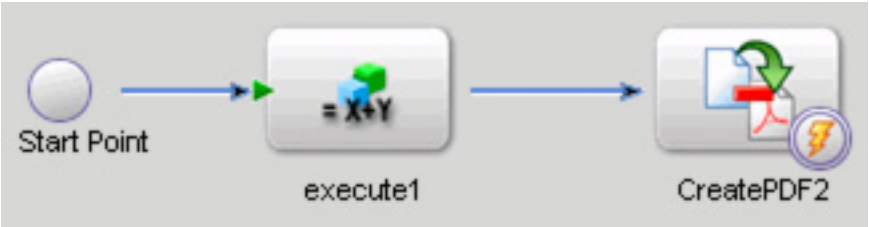
Catchingevent throws

Controllingprocess flow using events

Servicereference

## 11.6. Event start points

*Start points* are a special type of event receive that initiate short-lived and long-lived processes when events are thrown. Start points always react to events that are thrown. Start point events are similar to an event receive except that it starts listening for events as soon as the process is activated.




You configure *event filters* for start points that determine whether event throws are to be acted on. Event filters are based on the information in the event data. The event data and the event message data can be saved as process data.

A process diagram can have multiple start point events but only one of each type. For example, you can have only one TaskConsulted start point event and only one TaskEscalationTimer start point event. All start points are automatically connected with a route to the start activity of the process.

## Add event start points

1) Drag the event to an unused part of the process diagram.
2) (Optional) In the Name box, type a new name to replace the default name.
3) (Optional) To search for a specific event, replace the default string in the Find box with a value.
4) In the Event Behavior Configuration dialog box, ensure that Event StartPoint is selected.
5) (Optional) Add event filters. (See Addevent filters.)
6) (Optional) Store event data to process variables. (See Storeevent data to process variables.)

   ***NOTE:*** *The variables you store values to must be input variables.*
7) Click OK.

## Configure event start points

A start point event is similar to an event receive except that it starts listening for events as soon as the process is activated.

A process diagram can have multiple start point events but only one of each type. For example, you can have only one TaskConsulted start point event and only one TaskEscalationTimer start point event. All start points are automatically connected with a route to the start activity of the process.

**Configure an event as a start point:**

1) For an event throw or receive that is already on the process diagram, right-click the event and select Set Start Point. The Event Start Point Configuration dialog box appears.
2) In the Name box, type a name for the event that is meaningful and descriptive. This value is displayed in the process diagram.
3) On the Filter tab, create a filter for the event. (See Creatingevent filters.)
4) On the Callback Process Data Map tab, create a process data map for the event. (See Storingevent data to process variables.)
5) Click OK.

### Setting an event as the start activity

You can configure event throws as the start activity in your process to notify listeners that the process has started. For example, the start activity can send a notification to the server when your process is invoked.

If you want to start a process in reaction to an event that has occurred, use an event start point. (See Eventstart points.)

#### Set events as the start activity

- In the process diagram, right-click the event and select Set Start Activity.

RELATED LINKS:

Specifyingthe start activity of a process

## 11.7. Adding and defining abstract events

Add an abstract event to represent an event in the process when you are not sure which event to use. Abstract events are available from the process diagram Activity Toolbar. You can specify a name and description for abstract events to indicate the functionality that is required. At a later time, you define the event that provides the functionality. You can also delete abstract events that you no longer require.

You can change the event type that is defined for an abstract event at any time. However, if you change which event type is defined, the name and description that you have specified in the abstract event are removed.

Abstract events have no behavior at run time. You can test the execution of other operations in the process diagram while the abstract events remain undefined.

RELATED LINKS:

Deletingevents

### Add abstract events

1) Drag the abstract event to the process diagram.

### Define abstract events

1) Right-click the abstract activity and click Define Event.
2) In the Define Event dialog box, double-click the event type from the list, and then click one of the buttons to associate a behavior with the event.
3) Configure event properties.

RELATED LINKS:

Eventcategories

Receivingevent throws

Throwingevents

Eventstart points

# 11.8. Deleting events

You can delete events you no longer want in your process diagram or on an operation.

**NOTE:** You cannot delete exception event catches that are attached to operations by default.

### Delete events:

1) In the process diagram, delete the event by performing one of the following tasks:
   – Right-click the event and select Delete Event.
   – Select the event and in the toolbar, click Delete ✖, press the Delete key, or select Edit > Delete.

RELATED LINKS:

Controllingprocess flow using events

# 11.9. Mapping data for events

You can map data that is sent and received for events. The Map Data dialog box is used to configure event data for configuring event data and event message data. When you map data, you use a data schema that is provided defined by the custom event or service-specific event.

For an event throws you can configure the event data and optional event message data to send. (See Addingevent data to send.)

For event receives and event start points, you can create event filters, store event data, and store event message data in to process variables. (See Definingevent filtersand Storingevent data to process variables.)

### Adding event data to send

For throw events, you can configure the data that you want to send. The event data is sent is defined by the data schema for the event. System events are configured with a set of elements that you send when the event is thrown. Custom events types require that you provide XML Schema Definition (XSD) to define the element structure of the data you send.

When you map data to send, the expression consists of the following form:

```
expression1 operator expression2
```

- expression1 is an XPath expression that evaluates to a value in the process data tree.
- operation is always the equal sign (=).
- expression2 is an XPath expression that evaluates to a value in the event's source data set or a value in the process data tree.

In the following example mapping, the expression stores a credit score from an application form in the `creditscore` variable.

```
/process_data/@creditscore = /data/applicants/Applicant/creditscore
```

If you do not specify any process data map expressions, no values are configured for the event data that is sent with the event throw. In addition to event data, you can send addition event message data for an event throw using an `XML` variable. Typically, event message data is not meant to be altered and used as a document of record.

RELATED LINKS:

Storingevent data

Creatingevent filters

**Add event data and event message data**

1) If you are modifying an existing event receive, right-click an event receive and select Event Properties, otherwise, proceed to step 2.
2) In the Asynchronous Event Throw Configuration area, click the Add button.
3) In the Map Data dialog box, beside the Event Data box, click the ellipsis button.
4) In the Map Event dialog box, expand the element in the Event Data pane.
5) Double-click the element representing the event data you want to configure and click OK.
6) For the Event Data Value, select whether to assign the event data as a literal or XPath value:
   – LITERAL: In the Event Data box, type the value you want to assign to the event data.
   – XPATH: Click the ellipsis button beside the Event Data Value box. In the Asynchronous Event Throw Configuration dialog box, create the XPath expression and click OK.
7) Click OK.
8) (Optional) Repeat steps 2 – 7 to add additional event data. You can also click the Edit button to change existing event data or click Remove button to delete event data.

RELATED LINKS:

Eventtypes

Throwingevents

## Creating event filters

Event filters are based on the information in the event data. Both event data and event message data can be saved as process data.

Create event filters for event catches, receives, and start points to specify the conditions under which the event is acted on when the event occurs. An event filter specifies criteria on event data or event message data. An event filter can one or more filter expressions and all expressions must evaluate to True for an event to be acted on.

**NOTE:** If no filters are created for an event, the event is always acted on.

Event filters consist of expressions in the following form:

```
expression1 operator expression2
```

- `expression1` is an XPath expression that evaluates to a filter key or a value in the process data tree.
- `operator` is an arithmetic operator that you choose.
- `expression2` is an XPath expression that evaluates to a literal value or a value in the process data tree.

In the following example event filter, if the `creditscore` filter key contains a value greater than `800`, the filter resolves to `True`:

```
/data/applicants/Applicant/creditscore > 800
```

### Add event filters

1) If you are modifying an existing event receive, right-click an event receive and select Event Properties, otherwise, proceed to step 2.
2) In the Event Filter tab, click Add the button.
3) Click the ellipsis button beside the Event Filter box.
4) In the Map Event Content dialog box, expand the element under in the Event Data pane.
5) Double-click the element representing the event data you want to send and click OK.
6) In the Operator list, select an operator to specify the condition to use. For example, select the greater than (>) symbol to specify when your event data exceeds the value you configure in the Event Data box.
7) In the Event Data Value, you can select whether to assign the event data as a literal or XPath value:
   – LITERAL: In the Event Data box, type the value you want to assign to the event data.
   – XPATH: Click the ellipsis button beside the Event Data Value box. In the Asynchronous Event Throw Configuration dialog box, create the XPath expression and click OK.
8) Click OK.
9) (Optional) Repeat steps 2 – 8 to add additional filters and click OK.

### Edit an event filter

1) If you are modifying an existing event receive, right-click an event receive and select Event Properties, otherwise, proceed to step 2.
2) In the Event Filter tab, select an event filter and click the Edit button.
3) In the Map Data dialog box, modify the values for the Event Filter, Operator, and Event Data Value boxes, and click OK.
4) Repate steps 2-3 as necessary and click OK.

### Delete an event filter

1) If you are modifying an existing event receive, right-click an event receive and select Event Properties, otherwise, proceed to step 2.
2) In the Event Filter tab, select an event filter and click the Remove button.
3) Repeat step 2 as necessary and click OK.

RELATED LINKS:

Eventtypes

Receivingevent throws

Catchingevent throws

Eventstart points

## Storing event data to process variables

For event catches, receives, and start points, you can select data from an event and store it in process variables for use for a later step in the process. A *process data map* specifies the data that is stored when an event is acted on. You can store event data and event message data.

**NOTE:** For event start points, any data being mapped from an event to a process variable requires that the variable is configured as an input variable. (See Process inputand output data.)

Process data maps consist of expressions in the following form:

```
expression1 operator expression2
```

- `expression1` is an XPath expression that evaluates to a value in the process data tree.
- `operation` is always the equal sign (=).
- `expression2` is an XPath expression that evaluates to a value in the event's source data set or a value in the process data tree.

In the following example process data map, the expression stores a credit score from an application form in the `creditscore` variable.

```
/process_data/@creditscore = /data/applicants/Applicant/creditscore
```

A process data map can have multiple expressions. Each expression can map a different piece of data as required. If you do not specify any process data map expressions, no data is saved for later use in the process.

In addition to event data, you can send addition event message data for an event receive. The data that you receive is not typically altered during process execution because it is a document of record. The data can also be used by the process to validate the XML document structure.

### Add a process data map

1) If you are modifying an existing event receive, event catch, or event start point, right-click it and select Event Properties, otherwise, proceed to step 2.
2) In the Process Data Map tab, click Add button.
3) Click the ellipsis button beside the Process Data box.
4) In the Asynchronous Event Receive Configuration dialog box, create the XPath expression to represent the process variable to store the event message data to, and click OK.
5) Click the ellipsis button beside the Event Content box.
6) In the Event Content dialog box, double-click the element representing the event data or event message data to store to the process variable and click OK.
7) (Optional) Repeat steps 2 – 6 to add additional process data mappings. Click OK.

### Edit a process data map

1) If you are modifying an existing event receive, right-click an event receive and select Event Properties, otherwise, proceed to step 2.
2) In the Process Data Map tab, select a process data entry and click the Edit button.
3) In the Map Data dialog box, modify values in the Process Data and Event Content boxes, and click OK.
4) Repeat steps 2-3 as necessary and click OK.

### Delete a process data map

1) If you are modifying an existing event receive, right-click an event receive and select Event Properties, otherwise, proceed to step 2.
2) In the Process Data Map tab, select process data entry and click the Remove button.
3) Repeat step 2 as necessary and click OK.

RELATED LINKS:

Eventtypes

Receivingevent throws

Catchingevent throws

Eventstart points

# 12. Preserving process results using the Archive Wizard

You may need to preserve (archive) documents that the process creates (for example, the approved forms). The Archive wizard guides you through the steps of adding a set of archiving operations to your current process. These operations preserve (archive) a document in various ways, such as printing it or saving it to a file system. A document to be archived can be a form or another type of document, such as a spreadsheet. The wizard also provides the option of converting the document to a PDF before storing it.

Upon completion, the wizard creates a fully functional part of the process that contains operations that correspond to the selected archival methods. Properties for each operation and route in the process are set based on the options selected in the wizard. The wizard also creates all variables that are required to run the process.

## 12.1. To start the Archive wizard:

1) Drag the Archive wizard icon from the Activity toolbar to the process diagram.
   *The Archive wizard provides the following methods of archiving:*

**Email:**

The document is sent as an email attachment to the specified recipients. (See Archivewizard: Email server properties and Archivewizard: Email address properties.)

**Saving to a file system:**

The document is saved in the AEM Forms Server file system with the specified name. (See Archive-wizard: File system properties.)

**Saving to Content Store:**

The document is saved to the specified content store by using the specified node name. (See Archivewizard: Content services properties.)

**Printing:**

The document is prepared to be printed on the specified postscript printer. (See Archivewizard: Printer properties.)

You can select any combination of these archival methods. Depending on the selected methods, the wizard presents one or two panels per method, where you can provide detailed information. Only the panels that are related to the selected archival methods are included in the wizard.

If you do not have a required service installed, related methods are not available, as detailed in this table.

| Service not installed | Options not available |
|---|---|
| Generate PDF | Convert to PDF<br>Print to Printer |
| Assembler | Convert to PDF |
| Convert PDF | Print to Printer |
| Document Management | Save to Content Store |
| Output | Print to Printer |
| Forms | Convert to PDF<br>Convert to PDFA |

You can enter most of the settings either as a literal value or as an XPath expression:

– To enter a literal value, select *literal* from the list beside the setting. Then, either type the value, select it from the list, or click the folder icon and select the value from the listed applications.
– To configure an XPath expression, select *xpath* from the list beside the setting, click the folder icon, and build the expression. (See Creating XPathexpressions.)

**NOTE:** When a process created using Archive Wizard references a document that is located in a local application, after creating a new version of this process, the reference is broken. You must manually update the reference to the correct version of the document.

# 12.2. Archive wizard: Document and archival method properties

In the Document Selection and Archive Options panel, specify the document to be stored and the storing methods.

## Select a document to archive

Provide information about the document that you want to archive at the end of the process.

**Document:**

The document to archive. If you provide a literal value, this document must be part of an application. (See Working with Applications.)

**Form:**

The form to archive. If you provide a literal value, this form must be part of an application. (See Working with Applications.)

**Data:**

The variable that holds form data. If you already have this variable configured in your process, select it from the list. To configure a new variable, click the plus sign (+) icon beside the list and create a variable in the Variable dialog box. (See Creatingvariables.).

## Conversion options

Select the format for the archived document.

**PDF:**

The document is converted to PDF before it is archived. Selecting this option adds the invokeDDX operation to the process diagram. (See invokeDDX.)

**NOTE:** This option is not available if you selected Document and do not have the Generate PDF and Assembler services installed on the server, or if you selected Form and do not have the Output service installed on the server.

**PDFA:**

The document is archived in the PDF/A format. The PDF/A format is used for the long term archiving of electronic documents. The Assembler service supports the PDF/A format.

**None:**

The document is archived in its original format.

The PDF option is not available when either one of the following conditions is true:

- You selected the Document option, and you do not have the Generate PDF and Assembler services installed on the server.
- You selected the Form option, and you do not have the Output service installed on the server.

*The PDFA option is not available when either one of the following conditions is true:*

- You selected the Document option, and you do not have the Generate PDF and Assembler services installed on the server.
- You selected the Form data, and you do not have the Forms, Output, and Assembler services installed on the server.

## How do you want to archive the document?

Select any combination of the following storage options:

**Email:**

The document is sent by email to specified addresses. Selecting this option adds the sendWithDocument operation to the process diagram. (See SendWith Document.)

**Save to File System on Server:**

The document is saved at the specified location on the file system. Selecting this option adds the writeDocument operation to the process diagram. (See WriteDocument.)

**Save to Content Store:**

The document is saved in the specified content space. Selecting this option adds the storeContent operation to the process diagram. (See storeContent.)

**Print to Printer:**

The document is configured and printed on the specified postscript printer. Selecting this option adds the toPS2 and sendToPrinter operations to the process diagram. (See toPS2 and sendTo-Printer.)

## 12.3. Archive wizard: Email server properties

In the Email Server Settings panel, provide information required for connecting to an SMTP server.

**SMTP Host:**

The IP address or URL of the SMTP server (for example, localhost, or 127.0.0.1).

**SMTP Port:**

The port that is used to connect to the SMTP server.

**Sender Email Address:**

The email address that is associated with the email user account that is sending the email message.

**SMTP Authenticate:**

Whether the user authentication is required to connect to the SMTP server.

**SMTP User:**

The name of the user account that is required to log in to the SMTP server.

**SMTP User Password:**

The password that is associated with the SMTP user account.

**Security Protocol:**

The security protocol to use for connecting to the SMTP server:

**None:**

No protocol is used (data is sent in clear text).

**SSL:**

Secure Sockets Layer protocol is used.

**TLS:**

Transport Layer Security is used.

## 12.4. Archive wizard: Email address properties

In the Email Address Settings panel, provide email addresses for the email recipients. Also, configure the subject and text of the email message.

**NOTE:** Separate multiple email addresses with a comma.

**To:**

One or more email addresses to send the email message to.

**CC:**

One or more email addresses to send a copy of the email message to.

**BCC:**

One or more email addresses to send a copy of the email message to. The addresses are hidden from other recipients of the email message.

**Subject:**

The text to use for the email message subject.

**Message:**

The text of the email message.

**Attachment Name:**

The filename to use for the attachment. Include the filename extension if the email message receiver must associate the file with the software to open it.

## 12.5. Archive wizard: File system properties

In the File System Settings panel, specify the place in the AEM Forms Server file system where you want to save the document.

**File Path:**

A fully qualified path to the file location in the AEM Forms Server file system. The path must end with a forward slash (/) or a backslash (\) (for example, C:/Archives/).

**File Name:**

The name of the file to save.

## 12.6. Archive wizard: Printer properties

In the Printer Settings panel, provide information about the printer to use for printing the document.

**Printer Server URI:**

The URI of the print server to use. The AEM Forms Server must have access to the print server.

**Printer Name:**

The name of a particular printer on the specified print server.

## 12.7. Archive wizard: Summary

The last panel in the Archive wizard contains a summary of the selected configuration options. Review this summary carefully and, if necessary, use the Back button to return to the previous panels and correct any errors.

When you are sure that all information is correct, click Finish to create the process.

# 13. Designing human-centric processes

*Human-centric processes* involve one or more users in a business process. The service and tools that forms workflow provides enables users to participate in processes. For an overview of how to use the service and tools to design processes, see Involvingusers in processes.

The following steps summarize how to create human-centric processes:

1) Design how information is captured and presented to users.
2) Generate tasks and assign them to users.
3) Specify how the task can be used in Workspace.

After data is stored in variables, you can access or manipulate the data as required.

## 13.1. Involving users in processes

Processes that involve users typically require the users to review information or provide information, or both:

- The User service creates tasks that appear in users' To Do lists in Workspace. Users can review information that is presented in a form, Guide, or document, enter new information, attach files, add notes, and submit the task.
- Workspace endpoints enable users to open tasks from the Start Process page of Workspace and submit them to invoke a process.

If PDF forms are used, task forms can also be sent to users in email. Users can open and complete the form by using Acrobat Professional and Acrobat Standard or Adobe Reader.

When a user completes a task, all the information about the task is submitted to the AEM Forms Server:

- Information that was entered in the form or Guide
- Attachments and notes
- Task metadata, such as the task ID, when the task was assigned and completed, and who completed the task

When the information is saved in process variables, you can use it later in the process. For example, you can use submitted form data in another task so that a different user can act on it. Typically, information is also used to make routing decisions. You can use XPath expressions to retrieve an item of form data for use in a route condition. For example, in a purchase order example, the cost of the purchase is used to make routing decisions. Low-cost purchases are made automatically, and high-cost purchases require approval.

**NOTE:** The User service can be used only with Workspace and email. To use a different application for enabling users to receive, open, and submit forms, create a custom service for assigning tasks to users.

## Enable users to start processes

Workspace start points enable users to start processes from Workspace. When you add a Workspace start point to the process diagram, a process card appears on the Start Process page of Workspace. When users click the card, a form or Guide opens. When they submit the form or Guide, the process is started.

The properties of the start point determine the form or Guide that appears in Workspace. Also, you specify rendering properties, as well as which Workspace features can be used.

Workspace start points cause the creation of a corresponding TaskManager endpoint on the AEM Forms Server. When you save a process that includes a Workspace start point, endpoint properties that were modified using administration console are overwritten.

For information about how to add and configure Workspace start points, see Startingprocesses using start points.

## Send a task to one user

The Assign Task operation of the User service creates one task that is assigned to one user or group.

The Assign Task operation is useful when a succession of people capture or consume information. For example, an employee submits a purchase order request. The request is then routed to a person in the procurement department for review. For more information see AssignTask operation.

## Send tasks to multiple users simultaneously

The Assign Multiple Tasks operation of the User service creates several tasks that are assigned to multiple users or groups (or both) at the same time. The tasks are identical in that they all present the same information. You can create lists of users and use the lists to assign multiple tasks. User lists are useful when a specific group of people are involved in review activities across several processes.

Information that is submitted with each task is stored in a Task Result Collection variable. You can use XPath expressions to retrieve and process information from individual tasks.

The Assign Multiple Tasks operation is useful when a process requires that several people provide similar information. For example, at the end of each fiscal quarter, a process assigns a task to the vice president of each geographical sales group of your organization. To complete their tasks, each vice president attaches their quarterly sales report and then submits the task. The process retrieves each attachment from the collection of task results, which can be sent to the senior vice president at a later point.

Similarly, the Assign Multiple Tasks operation is useful when the process requires that several people review the same information, such as in document review and approval processes. For more information, see AssignMultiple Tasks operation.

## Document review and approval processes

A common business process requirement is that several users review information and provide feedback on it. The User service supports the review of information by multiple people in parallel or in series:

**Parallel:**

The Assign Multiple Tasks operation assigns tasks to several people simultaneously. The tasks require each user to review the same information.

For example, a change-management committee of a manufacturing company reviews proposals for improving existing products. The members of the committee vote to decide whether to approve or deny proposals. The Assign Multiple Tasks operation creates tasks that display the change proposal. Each committee member receives a task in their To Do list in Workspace, and either approves or denies the proposal.

**Series:**

Several Assign Task operations execute in series, and each operation assigns a task to a user. For each task, the results are used as input for the next Assign Task operation.

For example, in a purchase order process, large purchases are ultimately approved by executive-level managers. Before the executive reviews the request, it is first reviewed by one or more lower-level managers according to the organizational hierarchy. An Assign Task operation is added to the process diagram for each level of management that performs a review.

forms workflow provides the following features that are useful for review and approval processes. Some features are not available for both Assign Task and Assign Multiple Task operations.

### Workspace approval tools

When users open their task, they can add comments to it and see the comments that other reviewers added. Users can also see what action other users selected when they submitted their task.

### Collection data and XPath functions

The information that is submitted for each task of an Assign Multiple Tasks operation is saved in a collection variable called *Task Result Collection*. XPath functions can be used to evaluate the results. For example, you can determine how many people selected a specific action or what percentage of people

submitted the action. These functions are useful when assessing results of document reviews that occur in series. For more information see Assessing reviewand approval results.

### Completion policies

You can complete an Assign Multiple Tasks operation before all of the generated tasks are completed. This feature is useful when a decision can be made about a review without a response from every reviewer. For example, the acceptance of a proposal requires a majority of approvals from committee members. You can complete the Assign Multiple Tasks operation immediately after more than 50% of the tasks are completed when the Approve action is selected. See Adding completionpolicies to Assign Multiple Tasks operations.

## Electronic signature processes

Electronic signatures are useful for validating that a specific person has read a particular piece of information. Processes can incorporate electronic signatures that can be used with tasks. (See Signature.)

### Digital signatures

Digital signatures on PDF documents can be used to verify that a person has read a specific document:

- Users can be sent a PDF in a task that they sign with their own credential. When submitting the task, the document is submitted and remains unchanged to preserve the signature.
- The AEM Forms server can be configured to sign a PDF document that a user submitted.

### Click-through electronic signatures

Confirmation messages can be displayed to users in a dialog box when they submit a task. The message can be used to establish a contractual agreement.

## Alternate tools for interacting with processes

Although Workspace is the primary tool for interacting with processes, email messages and certain mobile devices can also be used.

### Email

In addition to receiving notification messages about new tasks and the occurrence of deadlines and reminders, users can complete tasks using email:

**Reply to task assignment notifications:**

You can include links that users can click to reply to task notification emails. (See Enabletask completion by replying to notification email.)

**Submit PDF forms:**

PDF forms for a task can be attached to task notification email messages. Users can open the form and submit it using email. (See Creating emailtemplates.)

### Mobile devices

Applications can be obtained for certain mobile devices for interacting with tasks. Users can complete tasks from their mobile device, and, for PDF forms, open the form for viewing:

- 
- For information about how to configure tasks for mobile devices, see Best practicesfor mobile devices.

# 13.2. Designing data capture and presentation

Involve people in processes to capture information from them, present information to them, or both. The following aspects of a AEM Forms application define the interface that enables users to provide or consume information:

**Assets:**

Assets, such as forms or Guides, that are presented to users.

**Data:**

Information that is used to populate the fields of assets that are presented to users.

**Action profiles:**

A set of services that defines the final form in which assets and data are presented to users.

When you create tasks, configure them to use assets, data, and action profiles. Before creating tasks, you must understand the types of assets that can be used to present data to users. You must also understand the different services that are used to process data and present it to users. With this information, you can decide whether to change default action profiles or create new ones.

## Assets for capturing and presenting data

You can use different types of assets for capturing data from users and displaying data to users. Generally, these assets contain fields for displaying items of data. AEM Forms supports Guides and several different types of forms and files for use in processes:

**Guide:**

A user interface based on Adobe Flash technology that steps users through the data-entry experience.

**Adobe PDF form:**

A PDF form that is designed using Designer.

**Adobe XML form:**

An XDP file that is designed using Designer. *Adobe XML forms* use the Forms service to render Adobe XML forms to HTML or PDF.

**Acrobat form:**

A PDF form that is created using Acrobat (or a similar tool).

**PDF document:**

PDF documents can be presented to users for reviewing information.

**Flex application:**

A SWF file that is created using Adobe (Deprecated)Flex Builder.

**TIP:** Third-party file types can also be presented to users. Users need the client software to open the file. For example, users need Microsoft Word installed to open Word documents.

Use the New Form wizard to create Adobe PDF forms and Adobe XML forms. The wizard configures the form automatically for use in processes and opens the form in Designer for developing it. (See CreatingForms.)

Use the New Guide wizard to create Guides and use the Guide Design editor to further develop it. See CreatingGuides.)

RELATED LINKS:

Aboutcaptured data

Aboutaction profiles

## About captured data

Data and assets appear as a single entity when they are presented to users. However, they can exist separately before and after users see them:

- Before users are presented with data, it is merged with an asset.
- When users submit tasks, the process stores the submitted data in variables.

When captured data is stored in a variable, you can pass the data from user to user at different steps in a process. You can modify all or some of the data as required, or extract values from the data to use for decision making. Storing data also enables you to archive it as necessary.

The separation of data and assets also enables you to use the same data with different forms or Guides. Furthermore, when maintaining applications in production, you can update the design of the form or Guide without altering the process or interrupting processes that are executing.

### Data formats

Forms and Guides are compatible with data of different formats. The format determines the type of variables to use to store the data that is captured with them.

| Media | Format of data | Variable type to use |
|---|---|---|
| Guide | XML data (in text format) | XML variable |

| Media | Format of data | Variable type to use |
|---|---|---|
| Adobe PDF form | Either PDF documents (in binary format) or XML data (in text format). | Depends on the format of the submitted data:<br>• Use document variables to store submitted PDF documents.<br>*Use XML variables to store submitted XML data. Configure the variable to store XDP data.* |
| Adobe XML form | Either PDF documents (in binary format) or XML data (in text format). | Depends on the format of the submitted data:<br>• Use document variables to store submitted PDF documents.<br>*Use XML variables to store submitted XML data. Configure the variable to store XDP data.* |
| Acrobat form | Either PDF documents (in binary format) or data in text format (FDF, XFDF, or HTML)<br>**NOTE:** *FDF, XFDF, and HTML data from Acrobat forms cannot be used directly with Adobe XML or Adobe PDF forms.* | Depends on the format of the submitted data:<br>• Use document variables to store submitted PDF documents.<br>*Use XML variables to store submitted FDF, XFDF, or HTML data.* |
| Flex application | XML data (in text format) | XML variable |
| PDF document | PDF documents (in binary format) | document variable |

**TIP:** Typically, forms are designed to submit PDF documents when they must preserve a digital signature on the form.

To specify the variable that stores submitted data, configure the Workspace start point, Assign Task operation, or an Assign Multiple Task operation.

### XDP form data

Adobe XML forms and Adobe PDF forms can submit field data in XML data package (XDP) format. XDP is XML that uses a particular structure and specific namespaces. The structure and use of namespaces are required when the data is used with XFA applications, such as the Forms service.

XML data that is retrieved from forms are enveloped with `xdp/datasets/data` elements. For example, the following XDP is field data that is submitted from an XDP form. `MortgateForm` is the root element of the form data:

```
<xdp:xdp xmlns:xdp="http://ns.adobe.com/xdp/">
<xfa:datasets xmlns:xfa="http://www.xfa.org/schema/xfa-data/1.0/">
<xfa:data>
<MortgageForm>
...
</xfa:data>
</xdp:xdp>
```

### Configuring XML variables for XDP data

You can configure XML variables so that they store XML data as XDP. The Save Data As XDP property is useful when saving XDP data for forms. For example, data that is submitted from a Guide is in XML data. To populate an XDP form with the same data, you need to add XDP-specific elements to the XML. To do so, you copy the XML data to an xml variable that is configured for XDP data.

Saving data s XDP provides the following benefits:

- You do not need to manually add elements to make the XML valid XDP. The variable is initialized with the XML elements, including namespaces, that XDP requires.
- You do not need to use XDP namespaces in XPath expressions when working with data in the XML variable. XPath expressions become namespace-unaware, regardless of any namespaces that are registered in the process.
- You can see the XDP nodes in XPath Builder so that the XPath expressions that you create use the correct path.

### Saving XML as XDP data

To save XML data in an XML variable that is configured for XDP data, use XPath expressions that include the XDP-specific elements. For example, the xml variable named *xmlVar* contains the following XML code:

```
<root>
<first_child>This is the first child of root</first_child>
</root>
```

A process needs to store the XML data in a variable named *xdpVar*, an xml variable that is configured for XDP data. The following XPath expressions are used with the Set Value service to save the XML in the xdpVar variable using valid XDP format:

```
/process_data/xdpVar/xdp/datasets/data = /process_data/xmlVar
```

The xdpVar variable now contains the following XML:

```
<xdp:xdp xmlns:xdp="http://ns.adobe.com/xdp/">
<xfa:datasets xmlns:xfa="http://www.xfa.org/schema/xfa-data/1.0/">
<xfa:data>
<first_child>This is the first child of root</first_child>
</xfa:data>
</xdp:xdp>
```

**NOTE:** xdpVar does not include the *root* element from xmlVar. The following expressions create the *root* element in xdpVar: /process_data/xdpVar/xdp/datasets/data/root = /process_data/xmlVar

*Overwriting the XDP elements*

Saving XML data directly in an xml variable that is configured for XDP data overwrites the XDP elements that the variable is initialized with. The following XPath expressions involve the xml variables of the previous examples:

```
/process_data/xdpVar = /process_data/xmlVar
```

The following XML code is the resulting content of the xdpVar variable:

```
<root>
<first_child>This is the first child of root</first_child>
</root>
```

*Configure an XML variable to save XDP data:*

1) In the Variables view, double-click the xml variable to open the Variable dialog box.
2) Select Store Form Data As XDP, and click OK.

**Displaying the xml data structure in XPath Builder**

Seeing the data structure of xml variables in XPath builder is useful when creating XPath expressions. (See XPathBuilder (Process Properties).)

To display the structure of xml variables, include the XML schema in the xml variable. Different types of assets can be imported from applications into XML variables to obtain the schema:

- An XML schema file.
- An XDP file that has an embedded XML schema.
- An Adobe data model. The schema is derived from the model.

**NOTE:** Assets must be checked in before you importing them.

Schemas are based on the XML Schema Definition (XSD) defined at http://www.w3.org/2001/XMLSchema.

*Specifying the root entity*

After you import an asset, you can specify which entity in the schema to use as root. The root is important for xml variables that are configured for storing XDP data. When the root node is specified, the initial value of the variable includes an element that represents the root node. For example, an xml variable has the following properties:

- The Store Form Data As XDP option is selected. When the option is selected, XML variables ignore prefixes registered for namespaces and XPath expresssions do not require the prefix. When deselected, you must use the registered prefixes for namespaces in the XPath expression.
- An XDP file is imported so that the root node is called MortgageApp.

The xml variable is initialized with the following XML:

```
<xdp:xdp xmlns:xdp="http://ns.adobe.com/xdp/">
<xfa:datasets xmlns:xfa="http://www.xfa.org/schema/xfa-data/1.0/">
<xfa:data>
<MortgageApp/>
</xfa:data>
</xfa:datasets>
</xdp:xdp>
```

**NOTE:** When you import an XDP file, the file does not need to include an embedded schema to identify the root node. Workbench can determine the root node of XDP files without the schema.

*Import an asset:*

1) In the Variables view, double-click an XML variable to open the Variable dialog box.
2) Click Import Asset.
3) Select the asset to import and click OK.
4) In the Root Entity list, select the root node.
5) In the Variable dialog box, click OK.

RELATED LINKS:

Aboutaction profiles

Modifyingand creating action profiles

Retrievingform field values

Changingvalues in form data

Usingdata models with form and Guide data

## About action profiles

Action profiles are used to specify how the assets and data are presented to users. They also specify how data that users submit is processed before being stored in process variables.

Action profiles define a set of services:

**Prepare data service:**

(Optional) Alters the data that is provided for use with the asset. This service is also used for prepopulating forms and Guides.

**Render service:**

Renders the data and the asset before it is sent to the user.

**Submit service:**

Processes the data that the user submits to the AEM Forms Server.

When PDF forms are used, action profiles can also specify how to extend the capabilities of the PDF when opened in Adobe Reader. For example, Adobe Reader users require extended PDF forms to submit them. (See ReaderExtensions .)

### Default action profiles

Each form and Guide includes a default action profile that is configured according to the type of the asset, such as a PDF or XDP file. The default action profile is also configured according to information that is provided in the New Form wizard and the New Guide wizard if they were used to create the asset.

You can create new action profiles or modify the default action profile if necessary. However, the default action profile is adequate for most use cases.

To specify the action profile to use for an asset, configure the Workspace start point, Assign Task operation, or Assign Multiple Tasks operation.

RELATED LINKS:

Modifyingand creating action profiles

Workspacestart point properties

AssignTask

AssignMultiple Tasks

## Modifying and creating action profiles

Modify or create an action profile to affect the way the associated asset and data are displayed to the user. Most often, modified or new action profiles are required in the following situations:

- The functionality of a prepare data service is required. (Default action profiles do not include a prepare data service).
- An asset is presented in different ways at different parts of the process.

For example, a form is used with several Assign Task operations of a process. Only the first operation requires that a prepare data service is executed. Therefore, the asset requires one action profile that uses a prepare data service, and another action profile that does not.

In addition to the prepare data, render, and submit services, you can specify how to extend PDF forms for Adobe Reader users. To use the Acrobat Reader DC extensions service, specify the Acrobat Reader DC extensions Rights credential to use to extend PDF forms. PDF forms are extended according to the properties of the credential that you select. The credential must already be added to Trust Store. For more information, see Managinglocal credentials in Administration Help.

The dialog box that you use to modify and create action profiles enables you to create minimal prepare data, render, and submit processes. You can further develop the processes at a later time.

**Modify or create an action profile:**

1) In the Applications view, right-click the asset and click Manage Action Profiles.
2) Select an existing action profile or create one:
   – To modify an action profile, click the name of the action profile in the list.
   – To create an action profile, click the plus sign button .
3) In the Name box, type a name. The name appears in the list of available action profiles in the properties of the Workspace start point or User service operations.

4) In the Prepare Data box, select, remove, or create a prepare data process:
   – To select a process, click the Select button .
   – To remove the selected process, click the Clear button .
   – To create a process, click the New Process button . (See Aboutprepare data services.)

5) If you are using default render services, in the Render Type area, select the format to present to users:
   – For Guides, select Form Guide and select the variation to use.
   – For Adobe PDF forms or Acrobat forms, select PDF.
   – For Adobe XML forms, select either HTML or PDF, depending on how you want to transform the XDP file.

6) If you selected PDF and you want to extend the form for Adobe Reader users, select Reader Extend, and then select the credential to use.

7) Check the Form Will Be Submitted Offline As A PDF checkbox. This enables users to submit the form offline using a supported version of either Adobe Reader or Adobe Acrobat without having to submit it by logging in to Workspace.

8) To specify a render process to use other than the default, click the Select button to select the process, or click the New Process button to create a render process. (See Aboutrender services.)

9) To specify a submit process to use other than the default, click the Select button to select the process, or click the New Process button to create a submit process. (See Aboutsubmit services.)

   **TIP:** To restore default values for all properties except the name and description, click Restore Default.

10) Click OK.

**Remove an action profile:**

1) Select the action profile and click the minus sign button .

**Creating minimal processes for action profiles**

When you create or change an action profile, you can create new processes to use for the prepare data service, render service, and submit service. When you create these processes, you can finalize the configuration of the action profile without having to close the Manage Action Profiles dialog box. When you finish configuring the action profile, you can close the Manage Action Profiles dialog box and then complete the development of the new processes.

New processes are given a default name and location with default variables that are required. You can add more variables if you need them.

*Prepare data process properties*

**Process name:**

The default value is *Asset_Name*PrepareData, where *Asset_Name* is the name of the asset that the action profile is associated with. You can change the name.

**Location:**

The default location is the location of the asset that the action profile is associated with. You can change the location.

**Process Variables:**

Two variables are created by default:

- taskContext is an input variable of type TaskContext. This variable holds information about the task that is using the asset and data, including the data to manipulate before rendering.
- xmlPrepareData is an output variable of type xml. This variable is used to return the data after it is manipulated.

*Render process properties*

**Process name:**

The default value is *Asset_Name*Render, where *Asset_Name* is the name of the asset that the action profile is associated with. You can change the name.

**Location:**

The default location is the location of the asset that the action profile is associated with. You can change the location.

**Process Variables:**

The following variables are created by default:

- taskContext is an input variable of type TaskContext. This variable holds information about the task that is using the asset and data, including the asset and data to render.
- applyRE is an input variable of type boolean that indicates whether to use the Acrobat Reader DC extensions service to apply usage rights for Adobe Reader users. The default value is false, which indicates no usage rights are applied. You can change the name and data type or delete this variable.
- credentialAlias is an input variable of type string that is used to specify the alias of the credential to use for applying usage rights. This property requires a value only if applyRE is true. You can change the name and data type or delete this variable.
- outFormDoc is an output variable of type document. This variable is used to return the rendered asset.
- runtimeMap is an output variable of type map.

*Submit process properties*

**Process name:**

The default value is *Asset_Name*Submit, where *Asset_Name* is the name of the asset that the action profile is associated with. You can change the name.

**Location:**

The default location is the location of the asset that the action profile is associated with. You can change the location.

**Process Variables:**

The following variables are created by default:

- taskContext is an input variable of type TaskContext. This variable holds information about the task that is using the asset and data, including the asset and data that is submitted. You can change the name.
- applyRE is an input variable of type boolean that indicates whether to use the Acrobat Reader DC extensions service to apply usage rights for Adobe Reader users. The default value is false, which indicates no usage rights are applied. You can change the name and data type or delete this variable.
- credentialAlias is an input variable of type string that is used to specify the alias of the credential to use for applying usage rights. This property requires a value only if applyRE is true. You can change the name and data type or delete this variable.
- outputDocument is an output variable of type document. This variable is used to return the rendered asset. You can change the name.
- runtimeMap is an output variable of type map.

*Create a prepare data, render, or submit process*

1) On the Manage Action Profiles dialog box, click the create process icon for the type of process to create:
   – To create a prepare data process, click the icon next to the Prepare Data Process box.
   – To create a render process, click the icon next to the Render Process box.
   – To create a submit process, click the icon next to the Submit Process box.
2) (Optional) To replace the default process name, type a new name in the Process Name box.
3) (Optional) To specify a different location than the default location, click the folder icon and select the location to save the process.
4) To add a variable, click the plus sign (+) icon .
5) To change the name of a variable, click the variable name and type a new name.
6) To change the data type of a variable, click the data type and select a different data type.
7) To remove a variable, select the variable and click the minus sign (-) icon .

RELATED LINKS:

Actionprofile services at run time

Aboutprepare data services

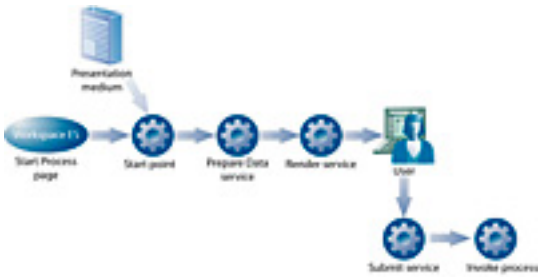
Aboutrender services

Aboutsubmit services

## Action profile services at run time

The services of an action profile are started when a user opens and submits a task. Users open tasks in Workspace in the following situations:

- They open a task from the Start Process page of Workspace. The Workspace start point for the process is executed to present the form or Guide.



- They open a form or Guide from their To Do list. The User service dictates which tasks appear in the To Do list.

*Similarly, the services are started when the AEM Forms Server sends a form to a user using email. In this situation, the user opens and submits the form by using a client application such as Adobe Reader.*

*For information about creating forms for forms workflow, see "Preparing Forms for AEM Forms forms workflow" in Designer Help.*

RELATED LINKS:

Involvingusers in processes

Creatingtasks

## About prepare data services

Prepare data services can be used to alter data before it is merged with an asset. Typically, prepare data services are that of an activated process.

For example, you are creating a process that prepopulates a form with a user's name, employee number, and other organizational information. You create a process that performs an LDAP query to retrieve the information. The process structures the data so that it is compatible with the form layout.

Because the way that data must be prepared is specific to your application or business process, create prepare data processes as you need them. AEM Forms does not provide default prepare data services.

If you create a prepare data process, it must include the following variables:

- A `document` or `xml` variable that has the Output property selected. The data in this variable is passed to the render service after the prepare data service is complete.

- A `TaskContext` variable that has the Input property selected. This variable contains information about the task, such as the identification of the user who is assigned the task.

You can also include a `map` variable as output. The contents of the map are appended to the `runtimeMap` data item of the `TaskContext` value. If you are also using a custom render service, that service can access the content of the map.

Prepare data processes can include any number of other input variables as you require. The values for the input variables are specified in the properties of the Workspace start point, Assign Task operation, or Assign Multiple Tasks operation.

### Advantages over subprocesses

You can use a subprocess to alter data before it is merged with the asset. The subprocess is started before the Assign Task or Assign Multiple Tasks operations are executed. However, no information about the task is available to the subprocess.

The prepare data service is started after the task is created so that the TaskContext value can be passed to the prepare data service. The information in the TaskContext value is useful for altering data based on the context of the task.

RELATED LINKS:

Creatingminimal processes for action profiles

Workspacestart point properties

AssignTask

AssignMultiple Tasks

## About render services

Render services merge data with forms or Guides before presenting them to users. For Adobe XML forms, render services also transform the form to a specific format, such as PDF or HTML. Typically, render services are services of an activated process.

forms workflow provides several default render processes that are deployed on the AEM Forms Server. The default action profiles of the different types of assets use these render processes.

It is unlikely that you need to alter the render processes that forms workflow provides or create custom render processes. However, if you must create custom render processes, they must include the following variables:

- A TaskContext variable that has the Input property selected.
- A document variable that has the Output property selected. The data in this variable must contain the rendered form or Guide when the service is complete.

Render processes can include as many input variables as you require. The values to use for the input variables are specified in the properties of the Workspace start point, Assign Task operation, or Assign Multiple Tasks operation.

### Rendering to HTML and sending email

When Workspace users are sent email notifications about new tasks, the task form can be attached to the email for filling offline. However, only PDF forms can be attached. If you are rendering forms to HTML for Workspace, the following task configurations cause a PDF to be sent to the user in email:

- The task is configured to send email notifications. However, if the task is only configured to send email notifications, then a link to the task in Workspace is sent out and not the PDF.
- The form data is included as a task attachment.

The form also requires specific configurations. The service that performs rendering must determine whether to render to HTML or to PDF. When forms are attached to email notification messages, the AEM Forms Server adds the following text to the targetURL field on the form:

```
mailto: [server address]
```

where *[server address]* is the email address that the AEM Forms Server uses.

If you create a render service, it must parse the content of the targetURL field to see if it contains the `mailto:` text:

- If the field contains the text, the service renders the form as PDF.
- If the field does not contain the text, the service renders the form as HTML. The form is not sent as a PDF attachment with the email notification.

RELATED LINKS:

Creatingminimal processes for action profiles

Workspacestart point properties

AssignTask

AssignMultiple Tasks

## About submit services

Submit services are used to process data after users submit their form or Guide. After the submit service processes the data, the data is stored in a process variable. Typically, submit services are services of an activated process.

forms workflow provides several submit processes that are deployed on the AEM Forms Server. The default action profiles of the different types of assets use these submit processes.

It is unlikely that you need to alter the submit processes that forms workflow provides or create custom submit processes. However, if you do create custom submit processes, they must include the following variables:

- A TaskContext variable that has the Input property selected.
- A document variable that has the Output property selected. The data in this variable contains the data from the form or Guide that was submitted.

Prepare data processes can include as many input variables as you require. The values to use for the input variables are specified in the properties of the Workspace start point, Assign Task operation, or Assign Multiple Tasks operation.

RELATED LINKS:

Creatingminimal processes for action profiles

Workspacestart point properties

AssignTask

AssignMultiple Tasks

# 13.3. Creating tasks

Create tasks to involve users in processes. Tasks are assigned to users and include a user interface (a form or a Guide) that can be populated with data. When users are assigned a task, they are provided with the form or Guide to complete and submit.

The following items can be added to process diagrams to generate tasks and involve users:

**Workspace start point:**

Enables users to open forms or Guides in the Start Process pages of Workspace.

**Assign Task operation:**

Creates a task that is assigned to a single user or user group.

**Assign Multiple Tasks operation:**

Creates tasks for multiple users so that they can simultaneously review information. The results of the reviews are stored in a Task Result collection.

After you add these items, specify values for presentation properties, such as the form and action profile to use. For Assign Task and Assign Multiple Tasks operations, also specify who is assigned the tasks that are generated. Workspace start point tasks are automatically assigned to the user who opened the form or Guide from the Start Process page.

When users submit forms and Guides to complete their tasks, the corresponding Workspace start point, Assign Task operation, or Assign Multiple Tasks operation is also complete. Data that populated the form or Guide is submitted to the AEM Forms Server. Data about the task is also submitted, such as who completed the task and which action was selected to complete the task.

For information about adding and configuring Workspace start points, see Startingprocesses using start points.

## Add an Assign Task or Assign Multiple Tasks operation:

1) Drag the Assign Task operation or Assign Multiple Task operation to the process diagram.

   *After you drag the operation to the process diagram, you can complete the following tasks:*

   – Configure the user to complete the task. (See Assigningtasks to users.)
   – Configure data capture experience for the user. (See Configuringthe presentation of task data.)

- Configure instructions to display to the user. (See Providingtask instructions.)
- Configure the actions for the user to complete a task. (See Providingactions for submitting tasks.)
- Configure the process to save the information details of the completed task. (See Savingtask data.)
- (Optional) Configure non-required task functionality. (See Configuringtask functionality.)

## Assigning tasks to users

For each Assign Task operation and Assign Multiple Tasks operation that you add to the process diagram, identify who is assigned the task:

- For Assign Task operations, specify a single user or a group.
- For Assign Multiple Tasks operations, specify multiple users.

You can use the following methods to identify a user:

**Specific user:**

Search for the AEM Forms user who always fulfills the role for this step in the process. If the user who fulfills the role changes, change the process accordingly. This method is useful for testing processes in the development environment. In the production environment, assign tasks based on roles or use XPath expressions.

**User list:**

(Assign Multiple Tasks operations only) Select a user list that has been created.

**Group:**

Assign the generated task to user groups so that tasks are distributed among the users in the group. Task assignment can occur in one of two ways:

- Tasks are assigned to a group's task queue, and users in the group manually claim tasks from the queue.
- Tasks are automatically assigned randomly to users in the group.

**Assign to process initiator:**

You can specify that tasks are assigned to the user who initiated the process without specifying the actual user profile.

**Variable:**

(Assign Multiple Tasks operations only) Select a variable that contains a User or Group value, or select a string value that stores an identifier for a user or group.

**XPath expression:**

Use an XPath expression that evaluates to a value that represents a user or group. XPath expressions are useful when the identification of the participant is stored as process data. For example, a previous task collects the identification in a form.

### Reassigning tasks for out-of-office users

Tasks can be automatically reassigned if the task is assigned to a user who is out of the office. Reassigning tasks prevents delays in the progression of the process.

Tasks are reassigned according to the Out Of Office settings that the user configured in Workspace. For example, the mortgage loan application that a bank uses requires loan approvals to occur in two days. A bank employee is on vacation and has configured the Out Of Office settings so that new tasks are reassigned to a colleague. When a task in the process is assigned to the employee who is on vacation, the task is automatically reassigned to the colleague.

**IMPORTANT:** You may not want to reassign tasks if they display confidential information to the user.

**Assign tasks for Assign Task operations:**

1) On the process diagram, select the Assign Task operation.
2) In the Process Properties view, expand the Initial User Selection property group.
3) Specify the user:
   – To select a user, select Assign To Specific User and then click Browse to search for the user. (See Selecta specific user).
   – To select the process initiator, select Assign To Process Initiator.
   – To specify a user from a group, select Assign To Group. Specify how tasks are assigned to users in the group, and then click Browse to search for the group. (See Selecta specific group.)
   – To use an XPath expression, select XPath Expression and then click the ellipsis button to open XPath Builder. (See Usean XPath expression or variable to specify users.)
4) To forward tasks according to users' Workspace out-of-office settings, select Allow Out Of Office Designation.

   **NOTE:** Tasks are forwarded only if the user's out-of-office settings indicate to do so.

**Assign tasks for Assign Multiple Tasks operations:**

1) On the process diagram, select the Assign Multiple Tasks operation.
2) In the Process Properties view, expand the Participants property group.
3) To add an item to the list, click the plus button . Select the type of item to add to the list, and then specify the specific item:
   – **User List:** In the Select User List dialog box, click the ellipsis button to search for the User List. (See Selecta user list.)
   – **User:** The Select User dialog box appears, where you can search for a user. (See Selecta specific user).

- **Group:** The Select Group dialog box appears, where you can search for a group. (See Selecta specific group.)
- **Variable:**The Select Variable dialog box appears, where you select an existing User, Group, or string variable that contains the identifier of a user or group. (See Usean XPath expression or variable to specify users.)
- **XPath Expression:** The XPath Builder dialog box appears. Create an XPath expression that evaluates to a User or Group value, or a string value that contains the identifier of a user or group. (See Usean XPath expression or variable to specify users.)

4) Repeat the previous step to add additional users or groups.
5) For each item in the Participants list, specify whether to forward tasks according to users' Work-space out-of-office settings:
   - Select the item in the list.
   - Select Allow Out Of Office Designation.
6) For each group in the list, specify whether a task is assigned to each member of the group:
   - Select the group.
   - Select Assign To Every User In The Group.

### Select a specific user

Search for the user to assign tasks that Assign Task or Assign Multiple Tasks operations generate. The following procedure describes how to use the Select User dialog box that appears when you are assigning tasks to specific users. For information about how to open the Select User dialog box, see Assigningtasks to users.

*Search for a user:*

1) In the Select User dialog box, specify how to search for the user:
   - To search by name, select User Name.
   - To search by the user's email address, select Email.
2) In the box, type all or part of the user name or email address and click Find.

   **TIP:** Type no characters in the box and click Find to retrieve a list of all users.
3) In the Results pane, select a user, and then click OK.

### Select a specific group

Search for the user group to assign tasks that Assign Task or Assign Multiple Tasks operations generate. The following procedure describes how to use the Select Group dialog box that opens when you are assigning tasks to specific groups. For information about how to open the Select Group dialog box, see Assigningtasks to users.)

Groups are defined by using User Management in Administration Console or the LDAP server defines them.

*Search for a group:*

1) In the Select Group dialog box, specify how to search for the group:
   – To search by name, select Group Name.
   – To search by the group's email address, select Email.
2) In the box, type all or part of the group name or email address and click Find.
   **TIP:** Type no characters in the box and click Find to retrieve a list of all groups.
3) In the Results pane, select a group, and then click OK.

### Select a user list

Browse for a user list to assign tasks that Assign Multiple Tasks operations generate. The following procedure describes how to use the Select User List dialog box that opens when you are assigning tasks to a user list. For information about how to open the Select User List dialog box, see Assigningtasks to users.

*Browse for a user list*

1) In the Select User List dialog box, click the ellipsis button to open a user list selection dialog box.
2) In the application tree, locate the user list to use to assign tasks.
3) Click the user list and click OK.

For information about creating a user list, see Creatingand changing user lists.

### Use an XPath expression or variable to specify users

You can use XPath expressions to identify users and groups that are assigned the tasks that Assign Task and Assign Multiple Tasks operations generate. (See Assigningtasks to users.) For Assign Multiple Tasks operations, you can also use variables. XPath expressions or variables are useful when the identification of the next participant is captured during the process.

The value obtained from the XPath expression or variable must be the identification of a user or group. If the identity of the user or group is not valid, an error occurs. The following information can be used to identify a user:

- Global unique identifier (GUID) of the user account, such as `9A7AD945-CA53-11D1-BBD0-0080C76670C0`
- Login name of the user, such as `atanaka`
- Canonical name of the user, such as `atanaka.sampleorganization.com`
- Email address, such as `atanaka@sampleorganization.com`
- Common name, such as `Akira Tanaka`
- The value that represents the user. See User for more information.

**NOTE:** Use the common name or email addresses only if you are certain that they are unique.

The following information can be used to identify a group:

- The GUID of the group, such as `C76670C0-CA53-11D1-9A7AD945BBD0-0080`
- The canonical name of the group, such as `tanakareports.sampleorganization.com`

- The email address, such as `tanakareports@sampleorganization.com`
- The common name, such as `TanakaReports`
- The value that represents the group. See Group for more information.

GUIDs are unique across all domains. Canonical names are guaranteed to be unique to a single domain only.

**TIP:** You can obtain User and Group values by using the User Manager Lookup service. (See UserLookup.*)*

### Creating and changing user lists

Create a user list to use for configuring task assignment for Assign Multiple Tasks operations. User lists are saved as application assets. You can edit or remove user lists any time after you create them. If you delete a user list, it is removed from the properties of any operations that refer to the list.

The New User List wizard steps you through the process of creating a user list. The wizard lets you specify the name and location of the user list, and the members of the list. After the user list is created, you can change which users are in the list.

*Start the New User List Wizard:*

1) Click File > New > User List.

   **TIP:** You can also right-click an application version and click New > User List.

*Change a user list:*

1) In the Applications view, right-click the user list and click Open.

   **TIP:** Ensure that you have the user list checked out.

RELATED LINKS:

Userlist general properties

Userlist population properties

### User list general properties

The following user list properties are configured by using the New User List wizard:

**Name:**

The name of the user list.

**Description:**

(Optional) A description of the user list that informs developers what the user list is used for.

**Location:**

The location to store the user list. Use one of the following methods to specify the location:

- Type the location in the box.
- Select the folder in which to store the user list in the application tree.

**TIP:** Click New Folder to create a folder to store the user list.

### User list population properties

You can configure the following user list properties by using the New User List wizard or when the user list is opened in Workbench:

**User List Population Options:**

Select an option to indicate how to specify the members of the user list:

- Select the users and groups to include in the user list to add the members manually. You can search for users and groups and add them to the list.
- Select use a process to populate the user list to run a process that selects the members.

The option that you select determines which properties appear.

*Select the users and groups to include in the user list:*

1) Select Users to add a user, or select Groups to add a group.
2) In the Name box, type part or all of the user or group name to search for, and then click Search.

   **TIP:** The search returns all users or groups when no name is entered.
3) In the list, select the user or group to add.
4) If you are adding a user, to forward the task based on the user's Workspace out of office settings, select Allow Out Of Office Designation.
5) If you are adding a group, specify how tasks are assigned to the group:
   – To create one task and add it to the group's To Do list in Workspace, select Assign To Group Queue.
   – To create a task for each member of the group, select Assign To Every User In The Group.
6) Click Add to add the user or group to the user list.
7) Repeat this procedure for every user or group that you want to add.

*Use a process to populate the user list:*

1) Specify the location of the process by using one of the following methods:
   – Type the location in the box.
   – Select the process in the application tree.

   *The process that you select must return a list value that contains the following data types (the list members must be the same data type):*
   – `string` values that represents any of the following user properties:
     - Global unique identifier (GUID) of the user account, such as `9A7AD945-CA53-11D1-BBD0-0080C76670C0`
     - Login name of the user, such as `atanaka`
     - Canonical name of the user, such as `atanaka.sampleorganization.com`
     - Email address, such as `atanaka@sampleorganization.com`

    - Common name, such as `Akira Tanaka`

  – User and group values. (See User and Group values.)

  *If a process for populating user lists is not yet created, click New Process. Workbench creates a minimal process that you can use to configure the user list properties. You can develop the process later.*

## Configuring the presentation of task data

For each Assign Task and Assign Multiple Task operation that you add to the process diagram, identify the information that is presented to the user. You can present an asset or a document that is stored in a document variable.

### Asset and data

When the Assign Task or Assign Multiple Task operation executes, the asset and data are sent to the assigned user. Specify which action profile of the asset to use. The prepare data, render, and submit processes of the action profile can require the following configuration:

- When to run the prepare data process
- Values for any input variables of the processes

### Document

Any document can be presented to the user if it is stored in the process data model as a document value. The user must have the client software required to open the document. Some applications require that the document opens in a new window.

**TIP:** Presenting PDF documents is useful when you need to preserve a digital signature that is on the document.

If you use a document variable, you cannot specify data to merge with the document.

### PDF assets with no submit buttons

Configure tasks so that Workspace provides submit buttons when all of the following circumstances are true:

- The task asset is a flat PDF document or a PDF form that does not include a submit button.
- Users open tasks using Adobe Reader version 9.1 or later, or Acrobat Professional and Acrobat Standard.

Under these circumstances, configure the Submit Via Reader property of Assign Task and Assign Multiple Tasks operations.

### Use an asset and data:

1) Select the Assign Task operation or Assign Multiple Tasks operation on the process diagram.
2) In the Process Properties view, expand the Presentation & Data Settings property group.
3) Select Use An Application Asset, click the ellipsis button, and select the asset.
4) In the Action Profile menu, select the action profile to use for the asset.

*If the processes of the action profile that you select require input values, corresponding properties appear at the bottom of the Application Asset area.*

5) (Optional) In the Variable menu of the Initial Task Data area, specify the process variable that contains data to populate the asset.
6) If you require Workspace to provide the submit button for the asset, select Submit Via Reader.
7) If you selected Submit Via Reader, in the list, select the type of data that you want to be submitted. The properties of the asset determine the available types. (See Aboutcaptured data.)
8) If the action profile that you selected in step 4 includes a prepare data process, specify when the process runs:
   – **The User Opens The Task:** When the user opens the task from their To Do list
   – **The User Opens A Draft Task:** When the user opens a draft task that they previously saved
   – **The User Opens A Completed Task:** When the user opens a task from their task history
9) If the processes of the action profile require input values, configure the remaining properties.

**Use a document variable:**

1) Select the Assign Task operation or Assign Multiple Tasks operation on the process diagram.
2) In the Process Properties view, expand the Presentation and Data Settings property group.
3) Select Use A Document Variable and then select the variable from the list.
4) If the document is a PDF document that Adobe Reader users will submit, select Submit Via Reader.
5) If you selected Submit Via Reader, in the list, select the type of data that you want to be submitted.

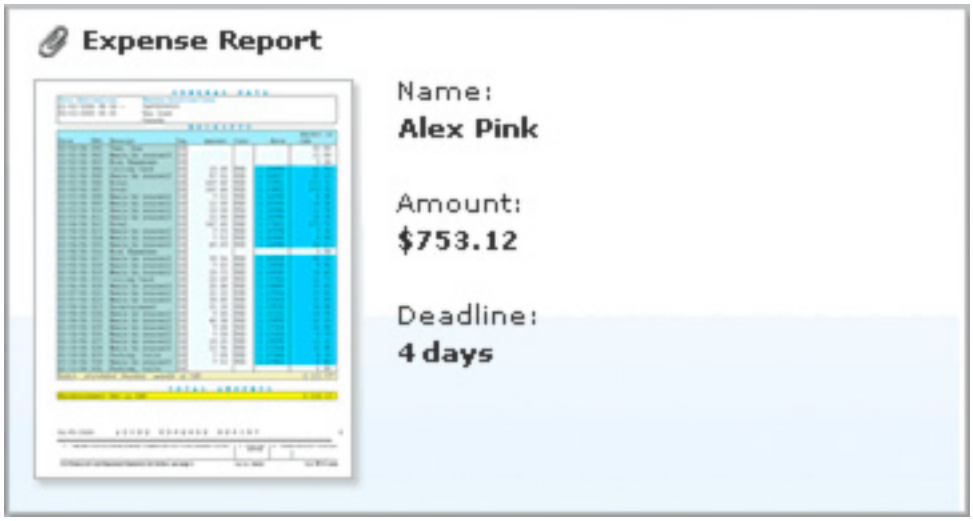
## Providing task instructions

Provide instructions to include with tasks so that users know what to do.

The text you provide appears on the process cards that appear in Workspace. Effective instructions can be created by expressing task instructions using a template value. You can also use HTML code to format the text. For example, the following text is a template value for the task instructions:

```
Name:<br><b>{$/process_data/@name$}</b></p>
Amount: <br><b>{$/process_data/@total$}</b></p>
Deadline:<br><b>{$/process_data/@deadlineDays$}</b></p>
```

The following task card in Workspace shows the resulting task instructions.




For more examples of effective layouts for task instructions, see Bestpractices for Workspace.

You can provide task instructions as a literal or template value, or you can specify a variable or XPath expression.

**Provide task instructions:**

1) Select the Assign Task or Assign Multiple Tasks operation on the process diagram.
2) In the Process Properties view, expand the Task Instructions property group.
3) In the Task Instructions box, specify the text to use for the instructions.

   **TIP:** You can also specify a variable that contains the text, or an XPath expression that evaluates to the text.

## Providing actions for submitting tasks

You can specify actions that users can select when they submit tasks. The actions you specify appear instead of the default Complete action. For example, the actions Approve and Deny can appear as buttons in Workspace.

The type of asset that is being used determines how actions appear in Workspace:

- If a Guide or PDF form is used, each action appears as a button. Clicking a button selects the option and completes the task.
- If an HTML form is used, each action appears in a list next to the submit button on the form itself rather than in Workspace. The user can select an item in the list before clicking the submit button.

**NOTE:** The form may require the Process Fields object to display the action options. (See “Preparing form designs for AEM Forms forms workflow” in Designer Help.)

The name of the action that the user selects is included in the submitted task data. The selected action can be accessed by using XPath expressions later in the process if necessary. However, XPath expressions are not necessary for selecting the route to follow after the Assign Task or Assign Multiple Tasks operations.

### Add a user action

1) Select the Assign Task or Assign Multiple Tasks operation on the process diagram.
2) In the Process Properties view, expand the User Actions property group.
3) Click the Add A User Action button .
4) In the Action Name box, type the name of the action as you want it to appear to the user.

***NOTE:*** *To express the value of the User Action Name property as a variable, the variable must be a* `list` *of* `string` *values.*

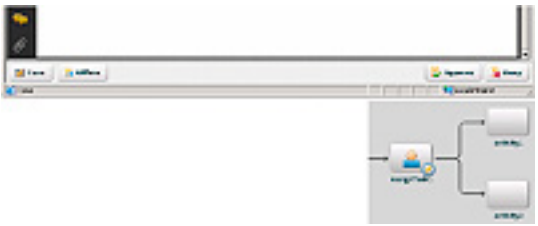
### Adding destinations for Assign Task operation actions

When you add user actions to Assign Task operations, you can associate each action with a subsequent operation. When the action is selected at run time, the route to the associated action is followed.

**NOTE:** Any conditions on the route are ignored.

If no operation is associated with the selected user action, the Route Evaluation properties of the operation and route conditions determine the next operation to execute.

For example, the following illustration shows the Approve and Deny actions. When the user clicks Approve, the route to activity1 is followed. When the user clicks Deny, the route to activity2 is followed.



Before you add destinations, draw a route to the desired destination.

*Specify a destination for an Assign Task operation:*

1) Select the Assign Task operation on the process diagram.
2) In the Process Properties view, expand the User Actions property group.
3) Select the action and click the Modify A User Action button .

   **TIP:** To add a user action, click the Add A User Action button .
4) From the Destination list, select next operation to execute when the user action is clicked, and then click OK.

**NOTE:** To express the value of the User Action Name property as a variable, the variable must be a `list` of `string` values. If you use a variable, you cannot associate user actions with routes.

RELATED LINKS:

Providingactions for submitting tasks

Requireconfirmation when submitting tasks

Addingand modifying routing conditions

### Adding completion policies to Assign Multiple Tasks operations

Create completion policies to complete Assign Multiple Tasks operations before all reviewers have submitted their task. Typically, the number of times a certain user action is clicked determines the outcome of the review. Often, decisions can be made before all reviewers submit feedback. For example, a majority vote only requires higher than 50% of reviewers' approval.

The completion policy is `Complete this step when more than 50 % pick the Approve action`.

Each time a user clicks an action to submit a task, completion policies are evaluated.

The Assign Multiple Tasks operation tracks the number of times each user action is selected at run time. Policies use this information to determine when the operation completes, thus terminating the review. Policies contain the following information:

- The name of the action to which the policy applies
- A threshold, which is either a percentage or number of users that select the action
- An operator for the threshold, which can be either *at least*, *exactly*, or *more than*.

For example, the following policy completes the operation when more than 50% of reviewers select the Approve action:

```
Complete this step when more than 50 % pick the Approve action
```

The action is Approve, the threshold is "50%" and the operator is "more than".

*Specify the next operation*

Policies can optionally specify the next operation to execute when the Assign Multiple Policies operation is complete. These policies are in the following format:

```
Go to operation_name when operator threshold pick the action_name action
```

The following policy completes the operation and executes the execute1 operation when more than 50% of the reviewers select the Approve action:

```
Go to execute1 when more than 50 % pick the Approve action
```

*Use route evaluation properties*

When policies do not specify the next operation to execute, the route evaluation properties of the operation and route conditions determine the next operation. These policies are in the following format:

```
Complete this step when operator threshold pick the action_name action
```

The following policy completes the operation when 50% of the users select the Approve action but does not determine the next operation to execute:

```
Complete this step when more than 50 % pick the Approve action
```

*Use the "exactly" operator carefully*

Using the "exactly" operator in a completion policy is risky when the threshold is based on a percentage. Depending on the number of reviewers, an exact percentage can be impossible to achieve. For example, a review that includes nine people cannot result in exactly half of the tasks having the same action clicked.

Use the "at least" operator to include an exact percentage in the completion policy. The following example policy causes the operation to complete when exactly 60% or more of all reviewers click the Reject action:

```
Complete this step when at least 50 % pick the Reject action
```

*Ensure early completion*

Create multiple completion policies to ensure that reviews are completed as soon as possible. When you create multiple completion policies, the Assign Multiple Tasks operation completes when either of the policies is true.

For example, a reviewed document is accepted when at least 50% of reviewers click the Approve action:

```
Go to publishDocument when at least 50 % pick the Accept action
```

The result of this rule is that the document is rejected when more than 50% of reviewers click the Reject action. However, when more than 50% of the reviewers click Reject, the preceding completion policy cannot be attained. The review completes only after all reviewers submit their feedback.

To ensure that the review is complete as soon as possible, a second policy is created:

```
Go to rejectDocument when more than 50 % pick the Reject action
```

The number of completion policies that you require depends on your review and the policies of your organization.

**TIP:** Before you create completion policies, add all user actions and draw routes to all of the next possible operations.

RELATED LINKS:

Providingactions for submitting tasks

Requireconfirmation when submitting tasks

### Creating completion policies

1) Select the Assign Muliple Task operation on the process diagram.
2) In the Process Properties view, expand the User Actions property group.
3) Select Use Completion Policies.
4) Click the Add A Completion Policy button .

   **NOTE:** In order to add a completion policy you must have first have a user action set up. See Adda user action for more information.
5) In the dialog box, specify how the next operation to execute is determined:
   – To specify the operation in the policy, select Go To A Specific Next Step.

- To use the route evaluation properties, select Use Routing Evaluation Order And Route Conditions To Determine Next Step.

6) Click the text to change the values:
   - If Go To A Specific Next Step is selected, click <some destination> and select the next operation.
   - To change the operator, click More Than and select the operator.
   - To specify the threshold, click <some value> and type a number. To change the threshold from a percentage to a count of users, click % and select User(s).
   - To specify the action that this policy is associated with, click <some action name> and select the user action.

7) Click OK.

## Saving task data

You can save the data that is sent to the AEM Forms Server when the user submits the form or Guide to complete their task. Saving the data is necessary if you want to use the data later in the process.

**NOTE:** If you save task data in a variable that already stores data, the existing data is replaced.

For Assign Task operations, task data is returned as a Task Result value. These values include information about the task, such as which user action was selected, the user who completed the task, task attachments, and user comments (if the Approval Container (Deprecated) features are used in Workspace). The field data from the form or Guide is also included. You can optionally store the field data separately in a document or XML variable, so that it is easy to pass to subsequent Assign Task or Assign Multiple Tasks operations.

Because Assign Task operations create a single task, save the submitted task data in a Task Result variable. Assign Multiple Tasks operations create several tasks. Therefore, the Task Result value from each task is saved in a special collection called *Task Result Collection*. Each Task Result value is appended to the Task Result Collection value. You can also append the output of an Assign Task operation to a Task Result Collection value.

### Task Result Collection and the Workspace Approval Container (Deprecated)

The Workspace Approval Container (Deprecated) displays information that is stored in the Task Result Collection that is used to store task results. Any comments and user action information that is stored in the Task Result Collection is displayed in the Approval Container (Deprecated).

For example, a process includes an Assign Multiple Tasks operation, followed by an Assign Task operation:

- Both operations use the Workspace Approval Container (Deprecated).
- Both operations use the same Task Result Collection variable to store task output.

The Assign Multiple Tasks operation creates tasks for several users so that they can review a document. As users complete their tasks, the results are stored in the Task Results Collection. When stored, they are immediately available to the Workspace Approval Container (Deprecated). Users can open their task in Workspace to see which actions were clicked for completed tasks.

The Assign Task operation executes when the Assign Multiple Tasks operation is complete. When the user opens their task, they too can see which actions were clicked for the tasks of the Assign Multiple Tasks operation.

**Save task data for Assign Task operations:**

1) Select the Assign Task operation on the process diagram.
2) In the Process Properties view, expand the Output property group.
3) In the Task Result box, specify the variable or location in the process data model to store the Task Result value.

   **NOTE:** If you save task data in a Task Result variable that already stores data, the existing data is replaced.
4) To store field data separately, in the Output Data box, specify the document or XML variable to store the field data.

   **NOTE:** If you save task data in a variable that already stores data, the existing data is replaced.
5) Specify whether to store the task data in a Task Result Collection variable:
   – In the Variable list, select the Task Result Collection variable.
   – To save field data in the collection, select Include Captured Data.
   – To save attachments in the collection, select Include Attachments.

   **NOTE:** Task data is appended to the Task Result Collection. No existing data is overwritten.

**Save task data for Assign Multiple Tasks operations:**

1) Select the Assign Multiple Tasks operation on the process diagram.
2) In the Process Properties view, expand the Output property group.
3) In the Variable list, select the Task Result Collection variable.
4) To save field data in the collection, select Include Captured Data.
5) To save attachments in the collection, select Include Attachments.

RELATED LINKS:

Specifyingthe Workspace user interface

Enableusers to start processes

Senda task to one user

Sendtasks to multiple users simultaneously

Configuringtask functionality

# 13.4. Configuring task functionality

You can configure Assign Task and Assign Multiple Tasks operations that the User service provides to control task functionality in Workspace:

- Enable users to complete tasks without opening them. (See Makeopening tasks optional.)

- Present a confirmation dialog box to users. (See Requireconfirmation when submitting tasks.)
- Specify the tool set that appears in Workspace. (See Specifyingthe Workspace user interface.)
- Provide reminders, set deadlines, and escalate tasks. (See Settingtime constraints.)
- Customize the text in task notifications and send tasks as email attachments. (See Overridingtask notification settings.)
- Restrict who a user can consult with or delegate a task to. (See Configuring taskdelegations and consultations.)
- Place limitations on who can be forwarded tasks or consulted with on tasks. (See Configuringthe sharing of forwarded tasks.)
- Create access control lists (ACLs) to restrict the features that specific users and groups can use. (See Configuringaccess to task functionality.)
- Configure how attachments and notes are used for the task. (See Configuring attachmentsand notes.)
- Assign a priority to the task. (See Specifyingtask priority.)

**NOTE:** Bestpractices for Workspace provides tips for maximizing the Workspace experience.

## Make opening tasks optional

By default, users are required to open tasks in Workspace before they complete the task. However for some processes opening the task to review the content is not mandatory. For example, the task description that appears in the To Do list can convey enough information that enables users to make decisions. In this situation, to work more efficiently, users can complete the task directly from the To Do list.

**NOTE:** Some processes require that users review the form before they complete the task. Some industries have common business processes where laws stipulate this requirement.

**Make opening tasks optional:**

1) Select the Assign Task or Assign Multiple Tasks operation on the process diagram.
2) In the Process Properties view, expand the Workspace User Interface property group.
3) Clear the Must Open The Form To Complete The Task property.

## Maximizing the form or Guide

You can cause task forms and Guides to use all available space in the web browser window by default when opened in Workspace. Users can still maximize or minimize the display area of the form as needed.

**Maximize the form or Guide**

1) Select the Assign Task or Assign Multiple Tasks operation on the process diagram.
2) In the Process Properties view, expand the Workspace User Interface property group.
3) Select Open The Form In Maximized Mode.

## Require confirmation when submitting tasks

You can display a confirmation message to Workspace users when they submit a task. Users can either click OK to agree to the message or click Cancel. The message can be any text that you provide. You can also include HTML code to format the appearance.

Confirmation messages are useful when a contractual agreement is required between you and the user. For example, your legal policies require that users attest to the validity of the information they provided in a form. When they submit the form in Workspace, the confirmation message can ask them to click OK to attest that the information they provided is true.

When users click OK in the confirmation dialog, the task completes. The Cancel button closes the dialog and returns the user to the task. Along with the IP address and the user ID that is also stored, the confirmation message is an effective click-through electronic signature.

Confirmation messages are associated with user actions. You can provide a different confirmation message for different user actions.

**Add a confirmation message to a user action:**

1) Select the Assign Task or Assign Multiple Tasks operation on the process diagram.
2) In the Process Properties view, expand the User Action Names property group.
3) Add a new action or edit the action to add the confirmation message to:
   – To add a new action, click the Add A User Action button .
   – To edit an action, click the Modify A User Action button .
4) In the Action Name box, type the name of the action as you want it to appear to the user.
5) Select This Action Needs Confirmation, And Will Use The Following Text As The Message.
6) In the box, type the message. You can include XPath expressions in the message. When evaluated at run time, the value of the expression is inserted into the message. To open XPath builder, click the ellipsis button .
7) Click OK.

## Specifying the Workspace user interface

You can control which set of tools are available in Workspace when users open tasks by selecting one of the following options:

**Default:**

The standard set of Workspace features are available.

**Approval Container (Deprecated):**

Provides tools for reviewing documents. The tools enable users to see the status of the review, add comments, and see the task instructions. This option is typically selected for review and approval processes.

**TIP:** The Task Result Collection that stores task output provides the information that appears in the Approval Container (Deprecated). (See Savingtask data.*)*

**Custom:**

You can develop a custom set of tools and integrate them with Workspace. The tools are created by using the AEM Forms SDK. (See Programming withAEM Forms.) For example, you can customize the way task notes and process variables are displayed, or create custom rules such as requiring attachments to be added before completing the task.

To use a custom set of tools, you also specify where the executable files are located on the AEM Forms Server.

**Specify Workspace tools:**

1) Select the Assign Task or Assign Multiple Tasks operation on the process diagram.
2) In the Process Properties view, expand the Workspace User Interface property group.
3) Select Default, Approval Container (Deprecated), or Custom.
4) If you selected Custom, click Browse and locate the SWF file for the custom tools.

## Setting time constraints

To ensure that tasks are completed within a specific amount of time, specify constraints on the amount of time a user has to complete tasks:

- Send reminders to the user as prompts to complete tasks. (See Sending remindersabout tasks.)
- Set deadlines and follow a specified route when the deadlines occur. (See Settingdeadlines for tasks.)
- Escalate tasks and send them to a different user after a specified period. (See Escalatingtasks.)

You can specify whether the time constraints that you configure are in calendar days or in business days. (See Aboutbusiness calendars.) Each user is associated with a business calendar, as configured by using the forms workflow administration tools. (See “Configuring Business Days” in forms workflow administration help

.)

**TIP:** Time constraints are important for Assign Multiple Tasks operations. There is a higher chance that delays can occur in completing all of the tasks that the operation generates.

RELATED LINKS:

Addingcompletion policies to Assign Multiple Tasks operations

### About business calendars

*Business calendars* define business and non-business days (for example, statutory holidays, weekends, and company shutdown days) for your organization. When using business calendars, AEM Forms skips non-business days when performing certain date calculations. You can specify whether to use business calendars for the following types of items:

- User-associated events such as task reminders, deadlines, and escalations
- Actions not associated with users, such as Timer Events and the Wait service

For example, a task reminder is configured to occur three business days after the task is assigned to a user. The task is assigned on Thursday. However, the following three days are not business days because Friday is a national holiday and the next two days are weekend days. The reminder is therefore sent on Wednesday of the next week.

**NOTE:** When you use business calendars, time calculations are done to the nearest day. When using the default calendar, calculations are done to the nearest minute.

When calculating dates and times using business calendars, AEM Forms uses the date and time of the server where it is running. The server does not adjust for the difference between time zones. For example, a task reminder is scheduled to occur at 10:00 A.M. on a server running in London. The user who receives the reminder is in New York City. The user receives the reminder at 5:00 A.M. local time.

You can define business calendars and business days by using the forms workflow administration tools in administration console. (See forms workflow administrationhelp.)

### Sending reminders about tasks

Set reminders for Assign Task and Assign Multiple Tasks operations to prompt users to complete the associated tasks. Reminders help to ensure that the user who is assigned a task completes it in a timely manner. You can send an initial reminder with the option of repeating the reminder at regular intervals.

The dates when reminders occur can be expressed by using either calendar days or business days. You can also change the task instructions when the reminder occurs. When a reminder occurs, a blue clock icon appears for the task in Workspace. You can also send an email message to the user.

**NOTE:** Reminders remain configured for Assign Task and Assign Multiple Tasks operations even if they are reassigned to another user through escalation. If you are configuring both reminders and escalations, ensure that the timing for them is appropriate.

The time is measured from when the task is first assigned to a user.

*Configure reminders:*

1) Select the Assign Task or Assign Multiple Tasks operation on the process diagram.
2) In the Process Properties view, click the All button to display the Reminders property group.
3) Expand the Reminders property group.
4) In the list next to the Reminders label, select Literal Value.

   *If you select XPath Expression, the expression that you create must evaluate to a ReminderInfo value.*

5) Select Enable First Reminder.
6) To calculate the date when the reminder occurs using business calendars, select Use Business Calendar.
7) Specify the amount of time that passes after the task is initially assigned until the reminder occurs:
   – In the Days box, type the number of days.
   – (Calendar days only) In the Hours box, type the number of hours in addition to the days you specified.
   – (Calendar days only) In the Minutes box, type the number of minutes in addition to the days and hours you specified.
8) To specify that the reminder occurs at regular intervals, select Enable Repeat Reminder.
9) To calculate the date when the reminder occurs using business calendars, select Use Business Calendar.
10) Specify the amount of time between reminders:
   – In the Days box, type the number of days.
   – (Calendar days only) In the Hours box, type the number of hours in addition to the days you specified.
   – (Calendar days only) In the Minutes box, type the number of minutes in addition to the days and hours you specified.
11) To change the task instructions when the reminder occurs, select Change Task Instructions On Reminder and, in the box, type new instructions. You can also click the ellipsis button and create an XPath expression that provides specific task instructions when the reminder occurs.

**Setting deadlines for tasks**

Set a deadline on a task if you want to limit the amount of time that users are given to complete the task. After the deadline period elapses, the task state is automatically set to complete and the task moves to the next step in the process. Optionally, the process can continue using a route that follows the task on the process diagram.

A deadline moves the task to the next step in the process, where as an escalated task remains on the same step in the process, but assigns that task to another user.

The date when a deadline occurs can be expressed by using either calendar days or business days. The time is measured from when the task is first assigned to a user. You can change the task instructions when the deadline occurs and a red clock icon appears for the task in Workspace. You can also send an email message to the user.

When a deadline occurs, if the user saved a draft of the task the saved information is submitted with the task. Typically, information that is saved with the draft is incomplete and not reliable. Follow a specific route when deadlines occur.

*Set a deadline:*

1) Select the Assign Task or Assign Multiple Tasks operation on the process diagram.
2) In the Process Properties view, click the All button to display the Deadlines property group.

3) Expand the Deadlines property group.
4) In the list next to the Deadline label, select Literal Value.

   *If you select XPath Expression, the expression that you create must evaluate to a DeadlineInfo value.*
5) Select Enable Deadline.
6) To calculate the date when the deadline occurs using business calendars, select Use Business Calendar.
7) Specify the amount of time that passes after the task is initially assigned until the deadline occurs:
   – In the Days box, type the number of days.
   – (Calendar days only) In the Hours box, type the number of hours in addition to the days you specified.
   – (Calendar days only) In the Minutes box, type the number of minutes in addition to the days and hours you specified.
8) To change the task instructions when the deadline occurs, select Change Task Instructions On Deadline and, in the box, type new instructions. You can also click the ellipsis button and create an Xpath expression that provides specific task instructions when the deadline occurs.
9) To follow a route after the deadline occurs, select Follow A Specific Route On Deadline and, in the Select Route list, select an existing route.

### Escalating tasks

For Assign Task operations, you can assign a task to another user when a task is not completed within a certain period. For example, if a loan officer does not complete a loan application within two days, it can be assigned to the loan officer's supervisor. Escalations are also useful for forwarded tasks.

**TIP:** You can also use reminders to encourage the completion of critical tasks.

To configure an escalation, specify the time period after which the escalation occurs, as well as the user who is assigned the task. The date when an escalation occurs can be expressed by using either calendar days or business days.

**IMPORTANT:** The process stalls if you escalate the task to a user who does not exist.

The escalation time period is measured from when the task is first assigned to a user. Escalation does not occur if you do not specify a time period.

*Reassign a task as part of escalation:*

1) Select the Assign Task operation on the process diagram.
2) In the Process Properties view, click the All button to display the Escalation property group.
3) Expand the Escalation property group.
4) In the Escalate Task area, select Escalate Task.
5) To calculate the date when the escalation occurs using business calendars, in the Schedule Escalation area, select Use Business Calendar.
6) Specify the amount of time after the task is initially assigned that the escalation occurs:
   – In the Days box, type the number of days.

- (Calendar days only) In the Hours box, type the number of hours in addition to the days you specified.
- (Calendar days only) In the Minutes box, type the number of minutes in addition to the days and hours you specified.

7) In the Select Escalation User area, specify the user who is assigned the task when the escalation occurs. (See Senda task to one user.)
8) Specify whether to reassign the task based on the user's out-of-office preferences. (See Assigning-tasks to users.)

## Overriding task notification settings

Email notifications can be sent to users when they are assigned a task, and when reminders and deadlines occur. Email notification settings are configured globally by using the forms workflow tools in administration console. However, you can override the following settings for tasks that Assign User operations generate:

- Specify that no notification is sent.
- Customize the body of the email message.

The settings that are configured in administration console are used by default. (See forms workflow administrationhelp.)

In Workspace, users can configure preferences so that they are sent email about tasks that they are assigned. By default, Workspace users are configured to receive email notifications. (See Workspace Help.)

**Configure email notifications:**

1) Select the Assign Task or Assign Multiple Tasks operation on the process diagram.
2) In the Process Properties view, click the All button to display the Custom Email Templates property group.
3) Expand the Custom Email Templates property group.
4) In the Custom Email Templates area, select the type of notification to configure:
   - **Task Assignment:**When the task is assigned to a user
   - **Reminder:**When a reminder occurs
   - **Deadline:**When a deadline occurs
5) Select the notification configuration to use:
   - **Use Server Default:**The notification settings are used as configured in administration console.
   - **Do Not Send Email:**No notifications are sent.
   - **Customize:**Send an email and use a custom email template to define the message subject and body.
6) If you selected Customize, click Edit Email Template to define the custom email template that is used. (See Creatingemail templates.)

7) Repeat steps 3–5 for each type of notification that you want to configure.

RELATED LINKS:

Enabletask completion by replying to notification email

### Creating email templates

Email templates define the subject and body of email notification messages. You can create email templates when you override the global settings for task notifications. (See Overridingtask notification settings.)

You can specify the following properties of email messages:

- Content of the message subject
- Content of the message body
- Encoding to use for the message
- Message format (text or HTML)
- Whether to attach the task form in the message. This option is not available for deadlines.

**NOTE:** Only PDF forms can be attached to email messages. If you are rendering XDP files to HTML for use in Workspace, the forms are rendered to PDF for email attachments.

Use the Email Template Editor dialog box to author the email template. The dialog box provides several buttons that simplify the authoring process. When you click a button, text is automatically added to the email template. The following table describes the buttons.

| Button | Description |
|---|---|
| XPath | Inserts an XPath expression enclosed by curly brackets ({}) and dollar signs ($). At run time, the value that the expression evaluates to replaces the expression. |
| @@ | Inserts forms workflow variables enclosed by @@ symbols. At run time, the value of the variables replaces the variable name:<br>**Task Id:**The identification of the current task.<br>**Instructions:**The value of the Task Instructions property of the Assign Task operation.<br>**Process Name:**The name of the process.<br>**NotificationHost:**The name of the computer that sends the notifications. In clustered environments, only one AEM Forms Server in the cluster sends notifications.<br>**Operation Name:**The value of the Name property for the Assign Task or Assign Multiple Tasks operation.<br>**Description:**The value of the Description property for the Assign Task or Assign Multiple Tasks operation.<br>**Actions:** A numbered list of user actions for the task. Users can click the action they want to use to complete the task. |

| Button | Description |
|---|---|
| URL (HTML only) | Inserts the HTML code that provides a link to the task:<br>`<a href="http://@@notification-host@@:<PORT>/workspace/Main.html?taskId=@@taskid@@">Click here to open the Task</a>`<br>***NOTE:*** *Type a port number in place of* `<PORT>` *in the URL.*<br>For information about other parameters that you can use in the URL, see WorkspaceURL parameters. |
| Para (HTML only) | Inserts`<p></p>` tags. |
| Break (HTML only) | Inserts `<br/>` tag. |
| Bold (HTML only) | Inserts `<b></b>` tag. |
| Italics (HTML only) | Inserts `<i></i>` tags. |
| Color (HTML only) | Inserts the following HTML code:<br>`<span style="color:Red;"></span>` |
| Font (HTML only) | Inserts the following HTML code for specifying a change in font style:<br>`<span style="font-family: arial, sans-serif; font-size: xx-small;">` |
| Link (HTML only) | Inserts the following HTML code for adding hypertext:<br>`<a target="_blank" href=""></a>` |
| Table (HTML only) | Adds the following HTML code, which defines a table that has two rows and two columns:<br>`<table border="1" cellpadding="1" cellspacing="0" width="100%">`<br>`<tr>`<br>`<td align="left" valign="middle" width="50%" bgcolor="Silver"> </td>`<br>`<td align="left" valign="middle" width="50%" bgcolor="Silver"> </td></tr>`<br>`<tr>`<br>`<td align="left" valign="middle" width="50%"> </td>`<br>`<td align="left" valign="middle" width="50%"> </td></tr>`<br>`</table>` |

**NOTE:** You can use other HTML tags as required.

For information about how to open the Email Template Editor dialog box, see Overridingtask notification settings.

*Configure a custom email template:*

1) In the Subject box, enter the content to use for the email subject. Use the XPath or @@ button as required.
2) In the Encoding list, select an encoding to use for the email.
3) In the Body Format list, select the format to use for the email message.
4) On the Body tab, enter the content to use for the email subject. Use the buttons to insert content as required.
5) (Task Assignment and reminders) To include the form and data as an attachment to an email for the task, select Include Form Data In Email.
6) (Optional) Click the HTML Preview tab to test the appearance of the email template and then click OK.

**Workspace URL parameters**

You can include parameters in the URL for Workspace so that hypertext links open to a specific area of Workspace. For example, if you are including the link to a task in a task notification email, the link can open the task in the web browser.

Workspace URLs have the following format:

`http://[server name]:[port]/workspace?[parm]=[value]`

*[server name]*is the name of the AEM Forms Server

*[port]* is the server port used for AEM Forms

*[parm]* is the name of the parameter

*[value]* is the value of the parameter

The following table lists the parameters and their valid values.

| Parameter name | Value | Result |
|---|---|---|
| `taskId` | The ID of an existing task. | Opens the form for the task in the To Do page. |
| `startPage` | The following values are valid:<br>• *`startProcess`*<br>*`toDo`*<br>*`tracking`* | Opens a specific Workspace page. The value determines the page:<br>**startProcess:** The favorite area of the Start Process page.<br>**toDo:** The To Do page.<br>**tracking:** The Tracking page |

| Parameter name | Value | Result |
|---|---|---|
| `startEndpoint` | *`[service name].[endpoint name]`*<br>• *`[service name]`* is the name of the service for a process version.<br>*`[endpoint name]` is the name of the Task Manager endpoint that has been created for the service.* | Opens the form in the Start Process page for invoking the process.<br>For example, the following URL opens the form for a process named SimpleMortgageLoan-PDF with a TaskManager endpoint named SimpleMortgageLoan-PDF:<br>`http://localhost:8080/workspace?startEndpoint=SimpleMortgageLoan-PDF.SimpleMortgageLoan-PDF` |

**TIP:** The ID of a task is not defined until the task is created. The ID is available when the prepare data service of an action profile executes. You can use the prepare data service to obtain the task ID. (See Aboutprepare data services.*)*

**Enable task completion by replying to notification email**

To complete tasks, users can reply to email notifications that they receive when they are assigned tasks. When replying, users need to include the name of a user action in the body of the reply. The included user action is used to complete the task.

You can include links in the task notification email that, when clicked, create the email message to use to reply. If no user actions are defined for the task, a single link is provided for completing the task. To include the links, add `@@actions@@` to the template for email notifications. (See Creatingemail templates.)

**TIP:** Include enough information in the notification email so that users can make decisions without opening the task.

The Assign Task or Assign Multiple Tasks operation must allow tasks to be completed without opening them. (See Makeopening tasks optional.) If opening tasks is required, replies to email notifications have no affect.

RELATED LINKS:

Alternatetools for interacting with processes

## Configuring task delegations and consultations

For Assign task operations, you can restrict who a user can delegate or consult with for a task:

- When users delegate a task, it is either forwarded to another user or shared. Another user who belongs to a specific group can claim shared tasks.
- When users consult a task, the task is assigned to another user to help complete the task. However, ownership still belongs to the user who consulted the task.

You can also restrict who a user can delegate or consult a task with by specifying a group of users.

**Restrict delegation or consultation of tasks:**

1) Select the Assign Task operation on the process diagram.
2) In the Process Properties view, click the All button to display the Reassignment Restrictions property group.
3) Expand the Reassignment Restrictions property group.
4) To restrict who can receive forwarded tasks and access shared tasks to the users in a group, perform the following steps:
   – Select Forward To And Share With Only Members Of This Group.
   – Click the Browse button, select a group, and click OK.
5) To restrict who can be used for consultation to the users in a group, perform the following steps:
   – Select Consult Only To Members Of This Group.
   – Click the Browse button, select a group, and click OK.

   **NOTE:** If you specify the property values by using an XPath Expression, the expression must evaluate to a Task Delegate And Consult data item.

## Configuring the sharing of forwarded tasks

By default, Workspace users can access tasks that are forwarded to queues that they shared queue access to. The Share Tasks For Shared Queues process, which is installed and activated when you install forms workflow, implements this feature.

For example, the director of a business unit configured the queue so that the business unit's administrator has shared queue access. When a Workspace user forwards the director a task, the Share Tasks For Shared Queues process is invoked. The administrator is also provided access to the forwarded task.

If you want to see forwarded tasks that are sent to shared queues, stall the Share Tasks For Shared Queues process.

**NOTE:** When the Queue Sharing service is deactivated, the associated user interface appears unchanged in Workspace. As a result, error reports can appear in the AEM Forms Server log.

RELATED LINKS:

Configuringthe sharing of claimed tasks

## Configuring the sharing of claimed tasks

By default, the Share Tasks For Shared Queues process allows Workspace users to access tasks that are forwarded to shared queues. You can also configure AEM Forms to allow users to access tasks that are claimed from other queues.

For example, user1 shares a work queue with user2, and user2 shares a work queue with user3. When user2 claims a task from the user1’s queue, it appears in the queue that both user2 and user3 share. Both users have access to it.

**NOTE:** When a user claims a task, Workspace locks the task. Users cannot see claimed tasks in To Do lists that they have shared access to.

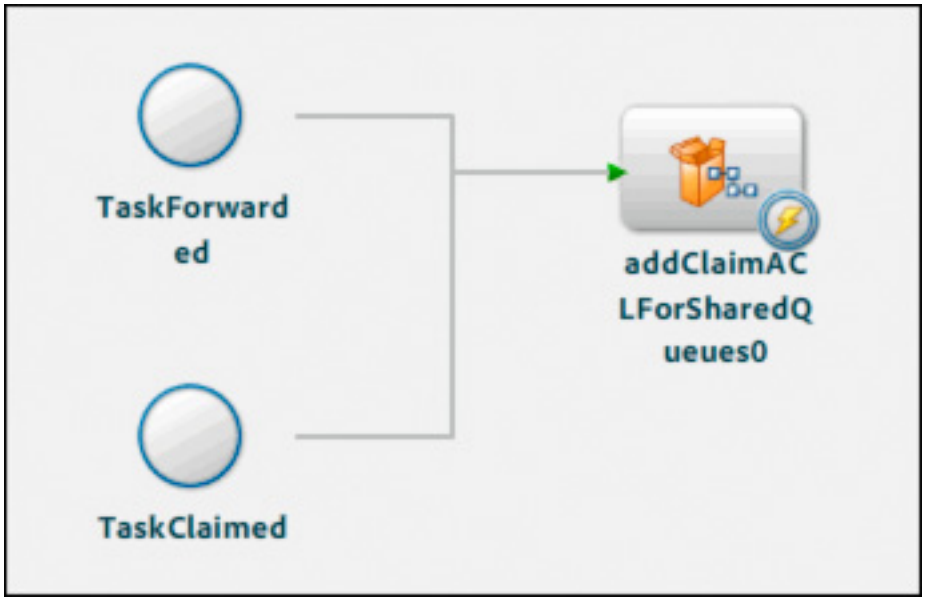
Because access to claimed tasks is not part of the installed Share Tasks For Shared Queues process, you must modify the process to add this functionality. This procedure involves adding a TaskClaimed event to the process diagram and configuring it to store the results in process variables. When this process is activated, users have shared access to both the tasks that are forwarded and the tasks that are claimed from another user's queue.

**Modify the Share Tasks For Shared Queues process:**

1) Click Window > Show View > Processes and expand the forms workflow category.
2) Import the Share Tasks For Shared Queues-1.0 process version in to a new application and check out the process.
3) Drag the Event abstract element onto the process diagram.
4) In the Name box, specify the name of the event (for example, TaskClaimed event).
5) Click Select An Event, expand the Asynchronous event category, select TaskClaimed, and click OK.
6) Click the Event Start Point and do the following tasks to configure the event:
   – Click the Process Data Map tab.
   – Click the elipses for the Process Data field and then double-click `taskID`. The following expression appears in the Process Data name box: `process_data/@taskID`. Click OK.
   – Click the elipses for the Event Content field, expand the MapContent tree, and then double-click `TaskID`. The following expression appears in the Event Content box: `/MapContent/TaskID`. Click OK.
   – Click OK to add the expression to the table.
   – Repeat the steps to create expressions that map `queueId` to `QueueId` and `userId` to `AssignedUser`.

   *The following expressions are in the table:*

   ```
   /process_data/@taskID = /MapContent/TaskID
   /process_data/@queueId = /MapContent/QueueId
   /process_data/@userId = /MapContent/AssignedUser
   ```
7) Click OK. The Share Tasks For Shared Queues process diagram looks like this illustration.




8) Save and check in the process .

## Configuring access to task functionality

The Access control list (ACL) restricts the use of Workspace features for specific users. For each user who can be assigned the task, you can create an ACL that controls the user's access to the following task features:

**Claim:**

The user can claim the task from another users queue.

**Add Notes:**

The user can add notes to the task and also set the access permissions for the notes.

**Share:**

The user can share the task. When a task is shared, the original permissions are enforced, and the user who the task is shared with can claim the task as their own.

**Forward:**

The user can forward the task to another user. When tasks are forwarded, the ACL is still enforced for the user it is forwarded to.

**Add Attachments:**

The user can add attachments to the task and also set the access permissions for the attachments.

**Consult:**

The user can consult the task. Consulting is similar to forwarding; however, the user who is consulted cannot complete the task. After the user who is consulted submits the form, it is returned to the original owner. The original owner then submits the form to complete the task.

### Default ACL

All Assign Task and Assign Multiple Tasks operations include a default ACL. Unless you create an ACL for a specific user, the default ACL is used for that user. To control the availability of features for specific users, create an ACL for that user. The default ACL allows users to add attachments, forward the task, and add notes. This ACL is suitable for most design scenarios.

The default ACL is named *<default ACL>*.

### ACLs and shared queues

You can extend the ACL to users who have shared access to the assigned user's queue. The permissions that are set in the ACL for the assigned user are extended to users with shared queue access.

**IMPORTANT:** Do not extend the ACL to users who have shared queue access if the task exposes confidential information. For example, users' social security numbers are typically kept private.

### ACLs and forwarded tasks

ACLs are not extended to users who are forwarded tasks. For example, when a user who has an ACL created for them forwards a task to another user, that user is restricted according to their ACL, or the default ACL.

### Specifying users

You can specify the users to apply ACLs to by searching for them or by using XPath expressions:

- Search for the user if the user is valid for all instances of the process.
- Use XPath expressions when the circumstances of the process instance determines who the ACL is created for.

When you use XPath expressions, the expressions must evaluate to one of the following types of information:

- The global unique identifier (GUID) of the user account, such as `9A7AD945-CA53-11D1-BBD0-0080C76670C0`
- The login name of the user, such as `atanaka`
- The canonical name of the user, such as `atanaka.sampleorganization.com`
- The email address, such as `atanaka@sampleorganization.com`
- The common name, such as `Akira Tanaka`
- The User value that represents the user

**Add an ACL for a user:**

1) Select the Assign Task or Assign Multiple Tasks operation on the process diagram.
2) In the Process Properties view, click the All button to display the Task Access Control List (ACL) property group.
3) Expand the Task Access Control List (ACL) property group.
4) In the Task ACLs area, click Add.
5) In the Search Type area, select the property of the user account to search on, either User Name or Email.
6) In the topmost box, enter text that matches all or part of the property that is being searched, and then click Find.
7) In the Results box, select the user, and then click OK.

   *You can add only one user at a time. When you add a user, their ACL provides access to the Add Notes, Forward, and Add Attachments features by default.*
8) Select the user from the user list.
9) Select and deselect permissions for the selected user.

**Extend the ACL to shared queues:**

1) In the Task Access Control List property group, select Add ACL For Shared Queue.

**Delete a user from the access control list:**

1) Select the user from the user list and click Delete.

## Configuring attachments and notes

You can specify whether the attachment window appears in Workspace for a task. You can also populate the attachment window with notes and attachments:

- Copy the notes and attachments that were attached to the previous task.
- Add the documents that are stored in a list variable as attachments.

When the task is completed, you can save the attachments and notes in a list of document variables.

**NOTE:** The document values are appended to the end of the list if the list value already contains documents. To replace a list, specify the index of the first item in the list. For example, you specify the XPath expression `/process_data/listVar[1]`.

Assign Task and Assign Multiple Tasks operations do not show the attachment window by default.When you use attachments or notes in a process, consider whether they are important to the process and how much storage space they require.

**TIP:** Attachments and notes within subprocesses are not directly available to the main process. You can make them available with an output variable of type `list` and a subtype of `document`.

**Configure attachments and notes:**

1) Select the Assign Task or Assign Multiple Tasks operation on the process diagram.
2) In the Process Properties view, click the All button to display the Attachments property group.
3) Expand the Attachments property group.
4) To show the attachment window in Workspace, select Show Attachment Window For This Task.
5) To specify the attachments to initially attach, use the Input List to select a list variable that contains the documents to attach.
6) For Assign Task operations, to save attachments that are submitted with the task, use the Output Attachments list to select a list variable to use to store the attachments.

   **TIP:** Attachments submitted for Assign Task and Assign Multiple Tasks operations can be saved with submitted task data. (See Savingtask data.*)*

## Specifying task priority

You can specify the priority of a task so that Workspace users can sort their tasks by priority. Tasks that have priorities other than NORMAL appear with color-coded exclamation marks. The following priority levels are provided:

- HIGHEST
- HIGH
- NORMAL
- LOW
- LOWEST

Specifying a priority does not affect how the process executes on the server.

**Specify task priority:**

1) Select the Assign Task or Assign Multiple Tasks operation on the process diagram.
2) In the Process Properties view, expand the Priority property group.
3) In the Task Priority list, select the priority to assign to the task.

RELATED LINKS:

Designingdata capture and presentation

Involvingusers in processes

Workingwith captured data

AssignMultiple Tasks

AssignTask

# 13.5. Working with captured data

This section provides information about how to use XPath expressions to work with task data that is submitted to the AEM Forms Server:

RELATED LINKS:

Assessingreview and approval results

Retrievingform data from XMl variables

Usingsubmitted data with Guides or Flex applicationsand forms

Documentattributes for attachments and notes

Addingdata nodes to xml variables

ConfiguringXML variables for XDP data

## Assessing review and approval results

You can use XPath expressions to assess the results of review and approval processes from Task Result Collection values. Assign Multiple Tasks operations enable you to create routes based on the results of document reviews. However, the XPath expressions are useful for the following situations:

- Sequential reviews where a series of Assign Task operations are used, and the results are accumulated in a Task Result Collection value.
- When routing decision need to be made sometime after an Assign Multiple Tasks operation.

For example, a succession of users reviewed a document and selected either the Approve or Reject user action. If a majority of reviewers approve, the document is accepted. After the final review, the process sends an email that includes the results of the review to all the participants. XPath expressions are used to determine how many times each action was selected.

Task Result values include the selectedUserAction data item that stores the user action that was used to complete the task. You can apply XPath functions to Task Result Collection values to determine the number of tasks that were completed by using a specific action:

**get-list-item-count:**

Returns the number of times an action was selected.

**get-list-item-percentage:**

Returns the percentage of tasks that a specific user action was selected.

These functions require an XPath expression that resolves to the Task Result Collection and the name of the action that you want information for. The following XPath expression returns the number of times the action named *Approve* was selected. The Assign Multiple Tasks results are stored in a Task Result Collection variable named *TRCvar*:

```
get-list-item-count(/process_data/TRCvar/object/taskResults,'Approve','s
electedUserAction’)
```

The function call includes `selectedUserAction` as the property name parameter. This optional parameter is included because the selectedUserAction data item of Task Result values stores the user action that was used.

For Task Result Collection values only, the get-list-item-count and get-list-item-percentage functions enable you to simplify the expression, Specify only the path to the Task Result Collection value, and do not provide a property:

```
get-list-item-count(/process_data/TRCvar,'Approve')
```

RELATED LINKS:

Documentreview and approval processes

Addingcompletion policies to Assign Multiple Tasks operations

Providingactions for submitting tasks

get-list-item-count

get-list-item-percentage

TaskResult

### Retrieving form data from XMl variables

Adobe XML forms and Adobe PDF forms can submit field data in XML data package (XDP) format. XDP.

XPath expressions are used for retrieving and setting form data that is stored in variables. To access the data, you must know how the data in the variables are represented internally.

The following process data model includes an XML variable named xmlVariable.

```
process_data
 ├ create_time
 ├ creator_id
 ├ id
 ├ status
 ├ update_time
 └ xmlVariable
   └ xdp
     └ datasets
       └ data
         └ MortgageApp
           ├ ApplicantFields
           ├ ApprovalFields
           └ MortgageFields
```

The root node of the data that is collected from the form is named MortgageApp. The structure of the data below this node represents the form schema. XDP data includes several levels of nodes above the root node of the form data.

The following XPath expression retrieves all of the form data:

```
/process_data/xmlVariable/xdp/datasets/data
```

## Using submitted data with Guides or Flex applications and forms

A common scenario involves collecting data using a Guide or Flex application, and then displaying the data to a different user in a form. It is also common to populate a Guide or Flex application with data that was submitted with a form.

Guides, flex applications and forms use XML data. Forms that are created using Designer require the XML to be in XDP format. However, Guide and Flex application data do not include the XDP-specific elements:

- To populate a form with the data that is submitted from a Guide or Flex application, you need to save the data in an XML variable that is configured to store data as XDP. (See ConfiguringXML variables for XDP data.)
- To populate a Guide or Flex application with data that is submitted from a form, you need to remove the XDP-specific elements from the XML. Use the Set Value service to extract the field data

and save it in another xml variable. Using the example XML data from Retrievingform data from XMl variables, the following Set Value mapping copies the field data from xmlVariable to xmlGuide, another xml variable:

```
/process_data/xmlGuide = /process_data/xmlVariable/xdp/datasets/data/*
```

## Document attributes for attachments and notes

When a file or note is attached to a task, a document value is created to represent the attachment or note. Several attributes are created for the document values that are used to store information about the attachment or note.

| Attribute name | Description | Default |
|---|---|---|
| `wsfilename` | The filename of the attached document or the title of the note that the document represents. The value can be the path to the file location and the filename, or just the filename. | `Attach`*`number`*, where *`number`* is the index that indicates the location of the document in a list variable used for storing the attachments and notes. |
| `wsdescription` | The description of the attachment or the note text | Empty string |
| `wscreatorid` | The user ID of the user who attached the document or provided a note to the task | Unique string that represents the user's identification |
| `wspermission` | A bit mask number that represents the access permissions for the document.<br>Access permissions determine how the document can be used when it is attached to a task. The value of the number is the sum of the number that represents each access permission provided for the document.<br>**`Read access:`** 1<br>**`Write access:`**2<br>`Delete access:` 4<br>Valid values are 1, 3, 5, and 7. For example, the value of `wspermission` for a document with read and write access is 3. | `7` (Read, write, and delete access) |
| `wsattachtype` | A `string` value that indicates whether the document represents an attachment or note:<br>• The value *`attachment`* indicates a binary file attachment.<br>*The value `note` indicates a note in text format.* | The attachment type always determines the value. |

You can use XPath expressions to retrieve or set the value of the attributes. (See getDocAttribute and setDocAttribute.)

For example, the notes and attachments of a task are saved in a list variable named `attachmentVar`. The following XPath expressions evaluate to the name, description, creator ID, and permission of the first attachment in the list:

```
getDocAttribute(/process_data/attachmentVar[1],"wsfilename")
getDocAttribute(/process_data/attachmentVar[1],"wsdescription")
getDocAttribute(/process_data/attachmentVar[1],"wscreatorid")
getDocAttribute(/process_data/attachmentVar[1],"wspermission")
```

The following XPath expressions replace the values of the name, description, creator ID, and permission for the first attachment in the list:

```
setDocAttribute(/process_data/attachmentVar[1],"wsfilename",
"BetterByAdobeRules.pdf”)
setDocAttribute(/process_data/attachmentVar[1],"wsdescription", "desc")
setDocAttribute(/process_data/attachmentVar[1],"wscreatorid","00ECB211-F
3A1-D93E-0EBE-EB13811A1C25")
setDocAttribute(/process_data/attachmentVar[1],"wspermission", "1")
```

## Adding data nodes to xml variables

You can add nodes to the data tree of xml variables and use them to store information during process execution.

**NOTE:** The added nodes do not appear in the data tree in XPath Builder.

Add nodes by using XPath expressions. When you use an expression that searches for a node that does not exist, AEM Forms creates the node when it evaluates the expression. After the node is created, you can use it the same way you use any other node in the tree.

The following rules apply when adding nodes:

- You can add nodes only beneath the form’s schema root node.
- Use the fully qualified path to the node.

For example, the following data tree is for a process that includes an XML variable named xmlLoanForm. The root element of the form schema is named MortgageSchema.

```
process_data
 ├create_time
 ├creator_id
 ├id
 ├status
 ├update_time
 └xmlLoanForm
   └xdp
     └datasets
       └data
         └MortgageSchema
           ├ApplicantFields
           ├ApprovalFields
           └MortgageFields
```

The following expression adds a node named `newNode` to the schema:

```
/process_data/xmlLoanForm/xdp/datasets/data/MortgageSchema/newNode
```

# 13.6. Best practices for Workspace

When developing human-centric processes for Workspace, process designers and administrators can employ several techniques to ensure a favorable user experience in Workspace:

- Use category names that appear properly in the Workspace navigation pane. (See Useshort category names.)
- Include an optimal number of process cards in each category. (See Limitthe number of process cards per category.)
- Format the content of card layouts so that users can quickly scan and absorb information. (See Useeffective card information.)
- Optimize the presentation of email notifications. (See Emailnotifications.)

## Use short category names

In Workspace, category names appear in the left navigation pane and are used to designate groups of processes. You create categories using Workspace start point properties.

**TIP:** You can also create categories using the Category Management area of Applications and Services in administration console. (See Applications andServices AdministrationHelp.*)*

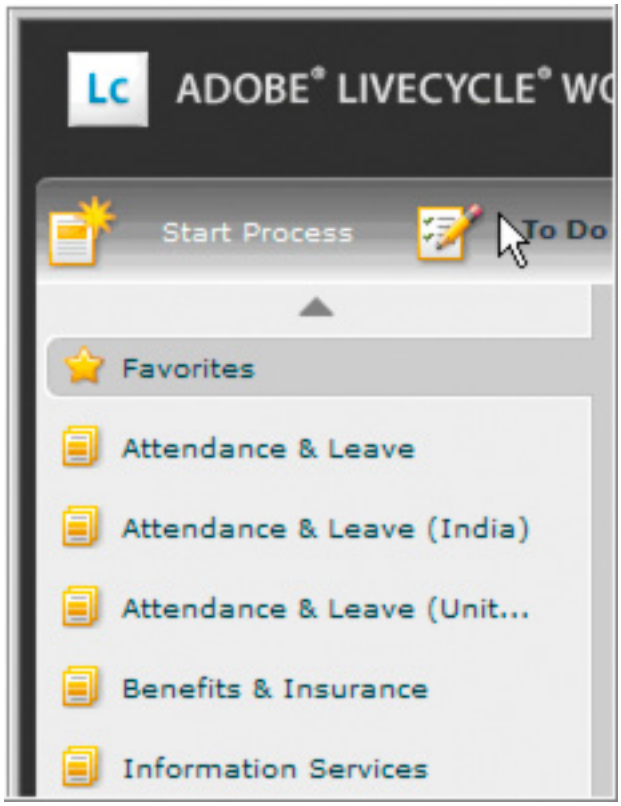
If the category name is too long to fit within the fixed width of the left navigation pane, it is truncated. The full name appears only when the mouse pointer is paused over it. By default, the navigation pane has a fixed width of 210 pixels, which is approximately 24 characters. Avoid using category names that are truncated:

**Category name that fits:**

Attendance & Leave

**Category name that is truncated:**

Attendance & Leave (United States)

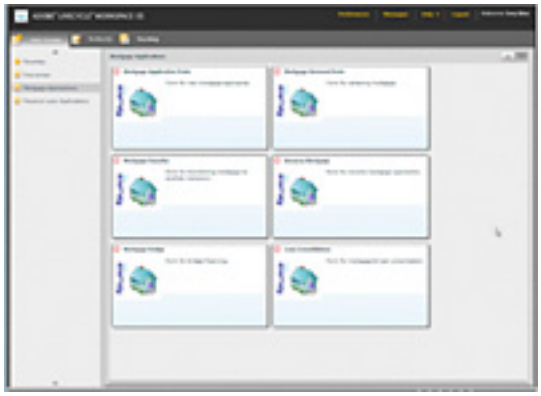



## Limit the number of process cards per category

In Workspace, cards on the Start Process page represent processes that users can invoke. You include processes in categories using the Category property of Workspace start points.

Limit the number of process cards per category so that users do not have to scroll the page to view cards. Cards that require scrolling in order to be seen are difficult to find. The resolution of the monitor affects the number of process cards that can fit on the screen without scrolling.

The following illustration shows the Start Process page for a category when the screen resolution is set to 1024 x 768.



At this resolution, the Start Process page can contain six process cards before scrolling is required.

RELATED LINKS:

Workspacestart point properties

## Use effective card information

Task cards in Workspace expose key information to users. Cards enable users to quickly scan tasks, make decisions based on information, and interact with processes without opening the task.

Cards include the following items:

RELATED LINKS:

Cardtitles

Taskinstructions on task cards

Imagesize in process cards

### Card titles

Use card titles that are descriptive and limited in length to prevent truncation:

**Card title that fits:**

Expense Report

**Card title that is truncated:**

Please approve this Expense Report

Keep the titles unique to help users locate the card quickly. Test the card title to ensure that it can be easily distinguished from other cards. Status icons that can appear to the left of the title can require titles to be shorter than you expect.

The way you specify card names depends on where the card appears:

- For cards on the Start Process page, use the Name property of Workspace start points. (See Workspacestart point properties.)
- For cards on To Do pages, use the Name property of the Assign Task or Assign Multiple Tasks operation. (See Commonoperation properties.)

### Task instructions on task cards

Provide instructions on process cards and task cards that are meaningful to users. On task cards, the values of process variables can be included to help users evaluate the task without opening it.

You can format task instructions by using HTML markup. For example, line breaks and the boldface type style are used to lay out the task instructions so that information can be easily scanned.

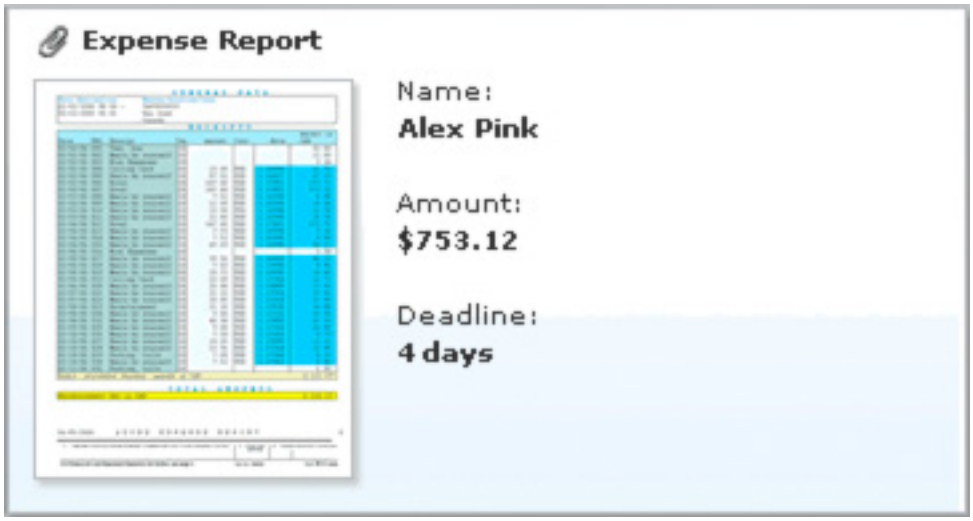
The way you specify task instructions depends on where the card appears:

- For cards on the Start Process page, use the Task Description property of Workspace start points. (See Workspacestart point properties.)
- For cards on To Do pages, use the Task Instructions property of the Assign Task or Assign Multiple Tasks operation. (See Providingtask instructions.)

*Instructions that use plain labels, line breaks, and bold variables*

The illustration below shows the following example task instructions:

```
Name:<br><b>{$/process_data/@name$}</b></p>
Amount: <br><b>{$/process_data/@total$}</b></p>
Deadline:<br><b>{$/process_data/@deadlineDays$}</b></p>
```

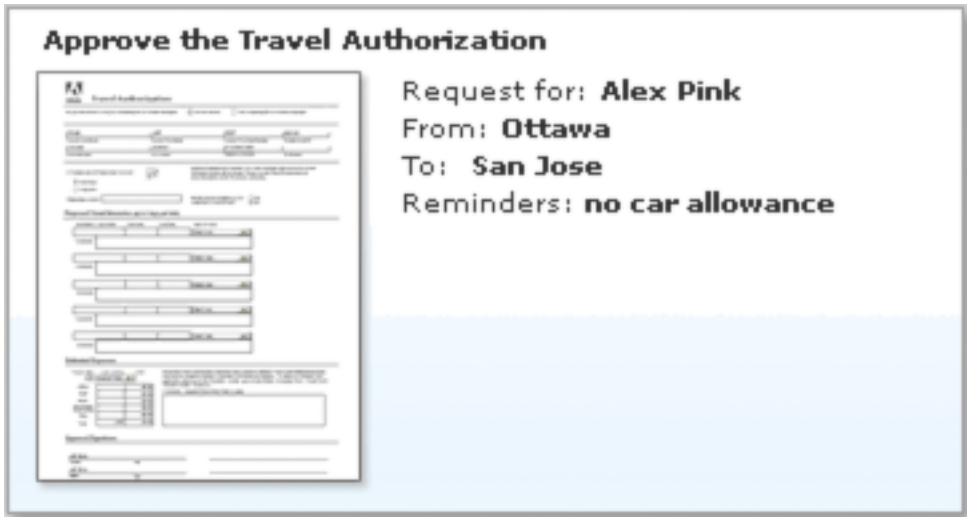



*Instructions that use plain labels, colons, bold variable, and line breaks*

The illustration below shows the following example task instructions.

**NOTE:** These instructions are contained in a single line of text.

```
Request for: <b>{$/process_data/@name$}</b><br>From:
<b>{$/process_data/@origin$}</b><br>To:
<b>{$/process_data/@destination$}</b><br>Reminders:
<b>{$/process_data/@hints$}</b>
```




*Instructions that use bold variables within static text*

The illustration below shows the following example task instructions.

**NOTE:** These instructions are contained in a single line of text.

```
<b>{$/process_data/@name$}</b> requested authorization for the
<b>{$/process_data/@processName$}</b> starting for
<b>{$/process_data/@total$}</b> in <b>{$/process_data/@quarter$}</b>.
Please verify that this request does not exceed your allowance for the
quarter.
```

### Image size in process cards

You can specify an image to be used that represents a process in Workspace. The image formats supported are JPG and PNG. The color should not be prescribed.

Reduce your image size to 48 x 48 pixels. You can compress the image before you add it to the process without losing quality. Use tools such as Adobe Fireworks® CS6, available from http://www.adobe.com.

## Email notifications

If your processes are designed to use email notifications, consider how the content of the email appears to the recipient. Task Instructions can be included in the body of the email by using the forms workflow variable @@instructions@@. If your email is configured to include the @@instructions@@ variable,

consider how that content appears when the email notification is sent to the recipient. If necessary, customize and format the email notification by using forms workflow variables and HTML tags.

Configure email settings in the forms workflow area in administration console. (See “Configuring notifications” in forms workflow Help.)

You can also configure email settings in Workbench. (See Overridingtask notification settings.)

### Submit button

Buttons associated with the submit buttons of the form are displayed in the Workspace chrome by default.

However, if you wish to hide the submit button, you can do the following:

When you are using a form as an Output Document variable, parameterize the`setDocAttribute` function as `setDocAttribute(document, “showWorkspaceSubmit”, “false”).` `showWorkspaceSubmit` is a document attribute that can be used with document functions and Render service.

## 13.7. Creating approval processes using Approval wizard

**** New for 9.5 ****

The Approval wizard guides you through the process of adding a User 2.0 operation for a document review and approval process. (See Documentreview and approval processes.)

Upon completion, the wizard adds a configured Assign Task or Assign Multiple Tasks operation (User2.0 service) to the process map. For serial reviews, a Set Value operation is added to the process map. Properties for the User 2.0 operation are set based on the options selected in the wizard. The wizard also creates all variables for the operation, which include:

- A xml variable to store the data that is merged with the form.
- A document variable to store the document that is sent for review and approval, which can be a PDF document or a form.
- A TaskResult output variable to access the results of the User 2.0 operation. (See Assessingreview and approval results.)

After the wizard completes, you can add variables, start points, and operations to complete your process map.

### Create a process using the Approval wizard

1) Drag the Approval wizard icon from the Activity toolbar to the process diagram.

The Approval wizard provides the following options to configure the task:

**Select A Form For Approval:**

You can choose to use a document variable or an existing form and its data to merge. The option you choose displays the form or PDF document to the user. (See Assetsfor capturing and presenting data.)

**Task Instruction:**

Provide instructions to the user to complete the task. You can provide information that sets the context for the review and approval. For example, you provide instructions that describe when to click the Accept or Decline button. (See Providingtask instructions.)

**How Do You Want To Route Your Approval:**

You can choose to send the approval to multiple users sequentially (Route In Sequence) or at the same time (Route In Parallel). (See Documentreview and approval processes.)

**Who Will Be Assigned To This Task:**

Selects the users to assign the task. You can select users & groups or a user list. (See Creatingand changing user lists.)

When you select a group, you can choose to assign the task to the group queue or each user in the group. When you assign the task to all users in a group, the task completes when one user completes the task. You can choose to allow out of office for the user or group.

**Specify Custom Names For Actions:**

You can provide custom names for the buttons displayed to users. If you do not provide a name for custom actions, the default Complete button is displayed to the user. (See Providingactions for submitting tasks.)

RELATED LINKS:

Bestpractices for Workspace

Configuringtask functionality

TaskResult Collection

Startingprocesses using start points

# 13.8. Best practices for mobile devices

Employ the following design practices when you expect users to open tasks from mobile devices:

- Configure the Assign Task or Assign Multiple Tasks operations to allow tasks to be completed without opening the form or Guide. (See Makeopening tasks optional.) On mobile devices, users cannot fill forms and cannot open Guides.
- Use Adobe PDF or Adobe XML (XDP) forms. Mobile users are sent PDF documents that are created from these types of forms for viewing:

  - If the Output service is installed on the AEM Forms Server, the form is flattened so that it is compatible with mobile devices. The PDF document is static and field values cannot be changed.
  - If the Output service is not installed, the PDF form is delivered.
  - If you use a different type of form, it is delivered to the mobile device. However, most devices support PDF.

- Use transparent 32x32 PNG images for the process icon. Larger images can be used, but 32x32 is ideal. The icon appears next to the task in the list of tasks. (See Creatingprocesses using the New Process wizard.)

RELATED LINKS:

Alternatetools for interacting with processes

# 13.9. Using certified signatures in forms in Workspace

When Adobe XML forms (XFA-based PDF forms) are used in processes that involve users with Workspace, the forms invalidate certified signatures. This is because when Workspace loads a static PDF, it breaks certification when the Submit button is hidden.

To use forms with certified signatures, perform the following steps when you create a process:

- Create a dynamic PDF by using an Adobe XML form and add the standard Process Fields to use the form in Workspace.
- Configure the `xfaForm` variable in your process to render the XDP file to a PDF. This requires that you configure the `xfaForm` variable with the Render PDF Form and Submit PDF Form services.
- Depending on the version of Adobe Reader or Acrobat you use, perform one these procedures:
  - For Adobe Reader 7 or Acrobat 7 and later, when using dynamic forms, modify the default Render PDF Form process. (See Modifyingthe Render PDF Form process to use certified signatures in a dynamicform with Workspace.)
  - For Adobe Reader 6 or Acrobat 6 and earlier, when using static forms, modify a setting on the form. (See Usingcertified forms with Acrobat or Adobe Reader version 6 or earlier.)

RELATED LINKS:

Modifyingthe Render PDF Form process to use certifiedsignatures in a dynamic form with Workspace

Usingcertified forms with Acrobat or Adobe Reader version6 or earlier

## Modifying the Render PDF Form process to use certified signatures in a dynamic form with Workspace

Use a modified copy of the Render PDF Form process that is installed with AEM Forms to enable the use of certified forms in Workspace. Modify the process when your organization uses Workspace and Adobe Reader 7 or Acrobat 7 and later to open PDF forms.

The Render PDF Form process is installed with AEM Forms. This process is used as the render process in the default action profiles of PDF and XDP forms. Do not modify this process directly. Create a copy, modify the copy, and change the action profile to use the modified copy.

**NOTE:** To use Adobe Reader, use the Acrobat Reader DC extensions service operation in your process to extend the usage rights for the form to include Enable Basic Form Fill-in and Enable Form Data Import/Export.

**IMPORTANT:** If a user digitally signs a form, to prevent changes to the form after it is signed, the form must be submitted to the AEM Forms Server as a PDF document. (See "To insert a submit button that embeds a PDF" in Designer Help.*)*

**To change the Render PDF Form process:**

1) Click File > Get Application, select forms workflow (system) > forms workflow (system)/1.0 > Render PDF Form, and click OK.
2) In the Applications view, right-click forms workflow (system) > forms workflow (system)/1.0 > Render PDF Form, and click Copy.
3) Right-click the application version to contain the modified process and click Paste.
4) Open the process and select the renderPDFForm operation.
5) In the Process Properties view, ensure the All button is pressed so that the properties are not filtered.
6) For the Acrobat Version property, select an Acrobat version that is Acrobat 7.0 or later. Select the earliest version that your users use.
7) Expand Render Options and for the Render At Client property, select Yes.
8) Save the process and deploy the Application.
9) Modify the action profile that you are using for your form so that the modified render process is used. (See Modifyingand creating action profiles.)

## Using certified forms with Acrobat or Adobe Reader version 6 or earlier

If your organization uses Adobe Reader 6 or Acrobat 6 or earlier, modify XML forms so that they can be certified. Modify a value in the process fields that you add to a form for use with Workspace. For the AWS_SUBMIT field, change the value of the Presence property from Visible to Hidden (Exclude From Layout). (See "Preparing form designs for AEM forms forms workflow" in Designer Help).

**IMPORTANT:** After you complete the procedure, the form cannot be used offline.

# 14. Using data models with processes

Data models created using Adobe application modelling technology are useful for passing data between application assets, and from Flex applications:

## 14.1. Using data models for process input and output variables

Data models can be configured so that when they are deployed, the data types that they define are available to processes. These custom data types simplify the integration of applications built using Flex technology (SWF files) and AEM Forms applications:

- Processes can include model-based variables as input or output data.
- Flex applications can provide values for the input variables when invoking processes.
- Flex applications can save the values that processes produce when completed.

  *Data models simplify the integration because their data types can comprise subtypes. For example, a custom data type is comprised of five string values. In the process, only one variable is required to store data of that type. Without the custom data type, five string variables are required.*

  *After the data is passed to the process, the process can use the data to populate forms or Guides, or for any other purpose. (See Usingdata models with form and Guide data.)*

  **NOTE:** Use XML variables to populate forms and Guides. Variables that store values of model-provided data types cannot be used to populate forms and Guides that are based on the same model.

  *Data types that contain subtypes behave in Workbench in the same way as other complex data types. Use XPath expressions to access data that the variables contain.*

  *To enable Flex applications to invoke a process, enable one or more programmatic end points for the Default start point. For example, to use the process web service to invoke the process, enable the SOAP end point.*

RELATED LINKS:

Starting processes using start points

## 14.2. Using data models with form and Guide data

Data models are useful at design time when you are working with XML data for forms and Guides that are based on data models. Model-based forms and Guides require XML data that is structured according to the same data model:

- To populate the form or Guide, you provide XML data that is structured accordingly.
- When you save form or Guide data in an XML variable, the data model dictates the XML structure.

  *You use the Set Value service to access data in XML variables. Data models can be imported into XML variables so that the model structure is displayed in XPath builder. In XPath Builder, seeing the model structure simplifies the creation of XPath expressions that you use to access the data.*

# 14.3. Integrating Data Models with Web Services and Data Services

Workbench enables process designers to use Data Models with external Web Services and Data Services. The new way is based on the visual data modeling using Workbench. Workbench generates a AEM Forms Component (DSC) for the data model, which contains create, read, update, and delete operations by default. The DSC is named after the data model and any configurations made to these service operations will be persisted with and made local to the data model.

## Integrating with Web Services

You can import a Web Service into a Data Model present inside of a AEM Forms Application and then generate classes to allow a Process to invoke the Web Service.

### Integrating with WSDL/SOAP-based Web Service using a Data Model

1) In Workbench, create a new Data Model using Web Services with an appropriate WSDL URL (**File** > **New** > **Data Model**).
2) In the Properties view, select the Generate LC Component option in the AEM Forms tab and save the data model.
3) In the Applications view, open the process where you wish to invoke the Web Service.
4) Select the Activity Picker and drag it on to a desired swimlane in your process map.
5) On the Define Activity panel, assign the activity one of the new service operations generated for your data model.
6) Provide Input and Output parameters in the Process Properties view for the activity.
7) Connect the new activity with other activities within the process.

### Integrating with WSDL-based Web Service using WSDL Service Reference

You can also integrate WSDL-based web services by adding a WSDL Service Reference:

1) In Workbench, create a new WSDL Service reference (**File** > **New** > **WSDL Service Reference**).
2) Select the Activity Picker and drag it to a desire swimlane in your process map.
3) On the Define Activity panel, assign the activity one of the new service operations generated for your data model.
4) Provide Input and Output parameters in the process properties view for the activity.
5) Connect the new activity with other activities within the process.

### Integrating with REST/HTTP-based Web Services

1) In Workbench, create a new Data Model based on HTTP service (**File** > **New** > **Data Model** > Provide a name on the **New Data Model** panel > Select **HTTP** on the **Select Service Type** panel).
2) In the Properties view, select the **Generate LC Component** option in the AEM Forms tab and save the data model.

3) Provide the return type for each operation in the Edit Signature dialog (select the operation in the Data Model window and click the Edit Signature button in the Properties view) and save the Data Model.
4) In the Applications view, open the process where you wish to invoke the Web Service.
5) Select the Activity Picker and drag it on to a desired swimlane in your process map.
6) Define the activity with one of the new service operations generated for your data model.
7) Provide Input and Output parameters in the Process Properties view for the activity.
8) Connect the new activity with other activities within the process.

# 14.4. Integrating with Data Services

Complete the following pre-requisite steps before you integrate a Data Model with Data Services:

1) Enable the RDS feature in the administration console by navigating to **Settings** > **Core System** > **Configurations** page and selecting the Enable RDS and Allow Non-secured RDS Request options.
2) Ensure that the database is accessible to the application server by registering it as a datasource.
3) Open the Preferences dialog in Workbench and update the RDS Configuration with the AEM Forms Server's hostname and port, "/remoting" as Context Root and AEM Forms Administrator's user-name and password.

## Integrating with an existing database

1) Create a new data model in Workbench with database by pointing to the appropriate datasource.
2) In the Properties view, select the **Generate LC Component** and **Deploy to LCDS** options in AEM Forms tab
3) Ensure that you select the Generate Service option in the AEM Forms tab in the properties view for all entities that you add to the model.
4) Save the data model and deploy the application.
5) In the Applications view, open the process where you wish to access the database.
6) Select the Activity Picker and drag it on to a desired swimlane in your process map.
7) Define the activity with one of the new service operations generated for your data model.
8) In the Process Properties view, provide Input and Output parameters for the activity.
9) Connect the new activity box with other activities within the process.

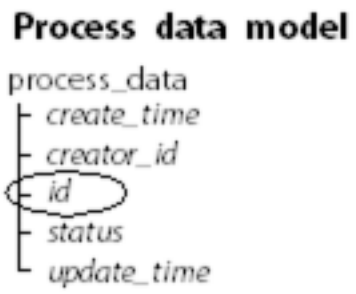
## Integrating with a new database

1) Create a new empty Data Model (**File** > **New** > **Data Model** > Provide a name and click **Next** > **Use an empty model as a starting point**).
2) In the Properties view, select the Generate LC Component and Deploy to LCDS options in AEM Forms tab

3) Ensure that new tables are created in the database by setting the Database Tables option to either **UPDATE** or **RECREATE**
4) In the Properties view, switch to the General tab and enter an appropriate JDBC Datasource name (e.g. enter “java:IDP_DS” to create new tables in AEM forms’s system database) and optionally select an appropriate Hibernate SQL Dialect (e.g. select “org.hibernate.dialect.MySQL5Dialect” when using Document Service Turnkey’s MySQL database).
5) Add new entities to the Data Model and ensure that you select the **Generate Service** option in the AEM Forms tab in the Properties view for all entities.
6) Save and deploy the Data Model.
7) In the Applications view, open the process where you wish to access the database.
8) Select the Activity Picker and drag it on to a desired swimlane in your process map.
9) Define the activity with one of the new service operations generated for your data model.
10) In the Process Properties view, provide Input and Output parameters for the activity.
11) Connect the new activity box with other activities within the process.

# 15. Creating XPath expressions

You can create XML Path (XPath) expressions in Workbench. XPath expressions are used in processes to identify data items in the process data model, such as data stored in process variables. For example, the XPath expression `/process_data/@id` identifies the `id` node in the process data model.




When this expression is evaluated at run time, it returns the data that the `id` node represents, which is the identification number for the process instance.

Expressions can include paths to data nodes, as well as functions and operators that act on the data:

- Simple expressions contain the path to a data item.
- Complex expressions apply functions or operators to one or more data items.

In processes, you can use XPath expressions to specify values for operation properties, values in configuration dialog boxes for events, and values in routing conditions. You can use XPath Builder to create expressions, or you can enter them manually. When you create XPath expressions manually, ensure that you use the same case as the variables you reference. Variables are case sensitive in Workbench.

For information about XPath, go to www.w3.org/TR/xpath in your web browser.

## 15.1. XPath Builder (Process Properties)

To build XPath expressions, you can use XPath Builder. XPath Builder includes the process data and common data trees, the function and operator libraries, and a work area where you create the expression:

- The process data tree, which appears in the uppermost pane of XPath Builder, shows the data model of the process in a tree structure. The data tree provides access to the information that is stored for the process, such as variable values and values in form data.
- The common data tree, which appears in the uppermost pane of XPath Builder shows the data-model of common variables in a tree structure. The data tree provides access to information that is stored as common variables and defaulted to use across all processes.
- The function and operator libraries, which appear in the uppermost pane of the XPath Builder, provide XPath functions and operators that you can apply to nodes in the data tree.

- The expression work area, which appears in the lower pane of the XPath Builder, is an editor where you author XPath expressions. When you double-click items in the data tree or the function and operator libraries, the items are inserted in the expression work area at the position of the cursor.

**A.**

Buttons for switching between Process Data tree, Functions library, and Operators library

**B.**

Process data tree. This box can also show functions and operators

**C.**

Expression work area.

**TIP:** Both functions and operations act on data values; however, functions act on parameter values, and operations do not. Operations act on values located before and after the operation; for example, the add operation in the expression `integer1 + integer2` acts on the values `integer1` and `integer2`.

You can open XPath Builder from the Process Properties view when the view is showing properties for routes and for operations, as well as from the dialog boxes for configuring events. An ellipsis button is provided beside text boxes where you can use XPath Builder to supply a value.

After you author the XPath expression and click OK in XPath Builder, the text in the expression work area populates the text box that is associated with the ellipsis button.

**TIP:** If you know the XPath language, you can enter expressions manually instead of using XPath Builder.

**TIP:** To open a list of values to complete the expression, press Ctrl+space. You can use the mouse or keyboard select an item in the list. Press Enter to insert the selection.

RELATED LINKS:

Editing text in the expression work area

Building simple XPath expressions

Building XPath expressions that use functions

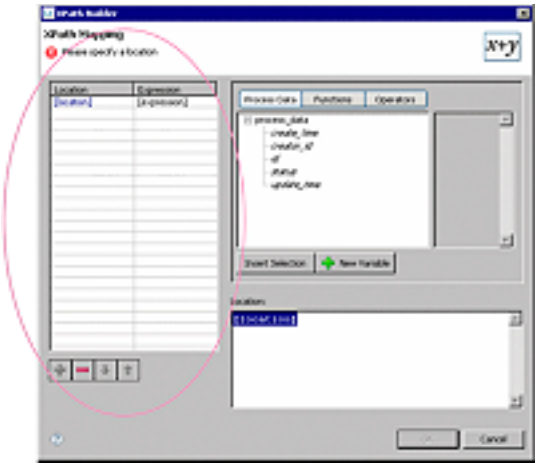
Building XPath expressions that use operators

XPath function reference

Example XPath expressions

# 15.2. XPath Builder (Data Mapping)

The XPath Builder for data mapping includes a mappings list in addition to the XPath authoring tools you use to define the locations and expressions. It allows you to edit the list for the Mappings property in the execute operation of the Set Value service.



## Location

The data item in the process data model that you want to set the value for. To specify the value, click the Location field and create the value using the XPath authoring tools. The value you create in the Location box is automatically added to the Location field. For information about the XPath authoring tools, see XPath Builder(Process Properties).

## Expression

The expression that defines the value for the data item specified for Location. To specify the value, click the Expression field and create the value using the XPath authoring tools. The value you create in the Expression box is automatically added to the Expression field. For information about the XPath authoring tools, see XPathBuilder (Process Properties).

**Add A List Entry:**

Click to add a mapping to the list.

**Delete Selected List Entry:**

Click to remove the selected mapping from the list.

**Move Selected List Entry Up One Row:**

Click to move the selected mapping up in the list. The list order determines the order in which mappings are executed at run time. The topmost mapping is executed first.

**Move Selected List Entry Down One Row:**

Click to move the selected mapping down in the list.

# 15.3. Building expressions by using XPath Builder

This section describes how to use XPath Builder to build expressions. The method you use to build an expression depends on whether you want a simple expression or a complex expression.

RELATED LINKS:

Editing text in the expression work area

Building simple XPath expressions

Building XPath expressions that use functions

Building XPath expressions that use operators

Creating XPath expressions

XPath Builder (Process Properties)

XPath function reference

## Editing text in the expression work area

The expression work area that XPath Builder provides is a simple text editor where you author XPath expressions. The expression work area supports many of the key combinations that are typically used with other text editors.

**To edit text in the expression work area:**

1) Click the work area to place the insertion point where you want to edit text, and use any of the following methods:
   - To add text, type text using the keyboard.
   - To paste text, press Ctrl+V.
   - To select text, drag the pointer across the text, or hold down the Shift key and use the arrow keys.
   - To delete text, press the Backspace or Delete key. To remove selected text and place it on the clipboard, press Ctrl+X.
   - To copy text, select the text and press Ctrl+C.

RELATED LINKS:

Building simple XPath expressions

Building XPath expressions that use functions

Building XPath expressions that use operators

Creating XPath expressions

XPath Builder (Process Properties)

XPath function reference

## Building simple XPath expressions

Build a simple expression when you must reference the data represented by a node in the process data tree. Simple expressions include a path to the data node.

You can also build expressions with nodes of XML variables that conform to an XSD that references XSD files.

In XPath expressions, the `@` symbol precedes simple variable types, and is not present in the complex variable types. The following expression references the value stored in the `string` variable named `stringVar`:

```
/process_data/@stringVar
```

The following expression references the value stored in the `xml` variable named `xmlVar`:

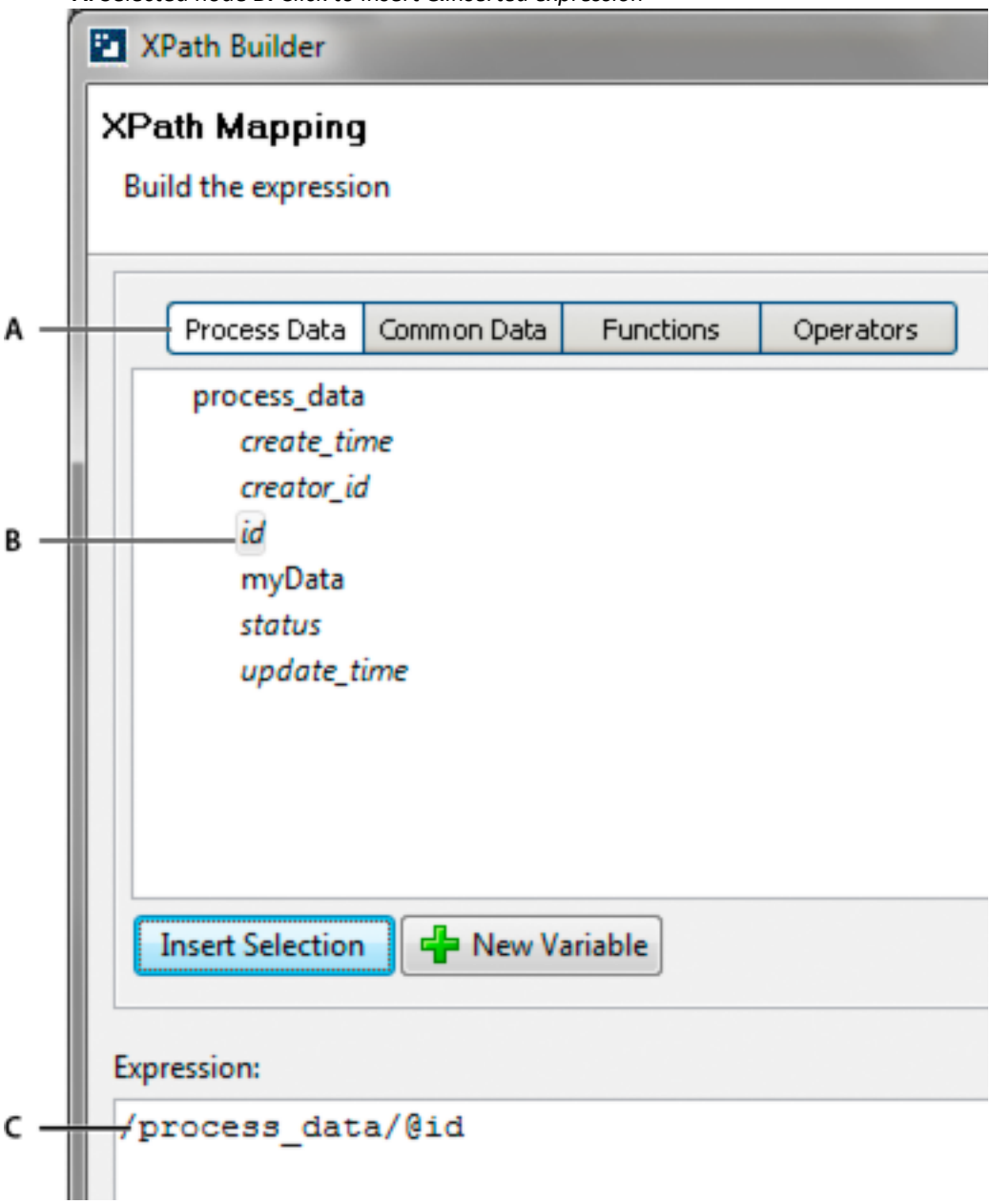
```
/process_data/xmlVar
```

**To build an expression that uses process variables:**

1) Navigate the process data tree to locate the item that you want to include in the expression.
2) Choose either of these ways to add the item to the expression:
   – Select the item and click Insert Selection.
   – Double-click the item.

   ***NOTE:*** *To build expressions that contain nodes of XML variables that conform to an XSD that references XSD files, drill down to the nodes as desired in the XPATH builder.For example, if you select the `status` node and click Insert Selection, the expression that identifies that node appears in the expression work area.*

**A.** Selected node **B.** Click to insert **C.***Inserted expression*




3) Repeat step 2 as required.
4) When the expression is complete, click OK.

RELATED LINKS:

Editing text in the expression work area

Building XPath expressions that use functions

Building XPath expressions that use operators

Creating XPath expressions

XPath Builder (Process Properties)

XPath function reference

## Building XPath expressions that use functions

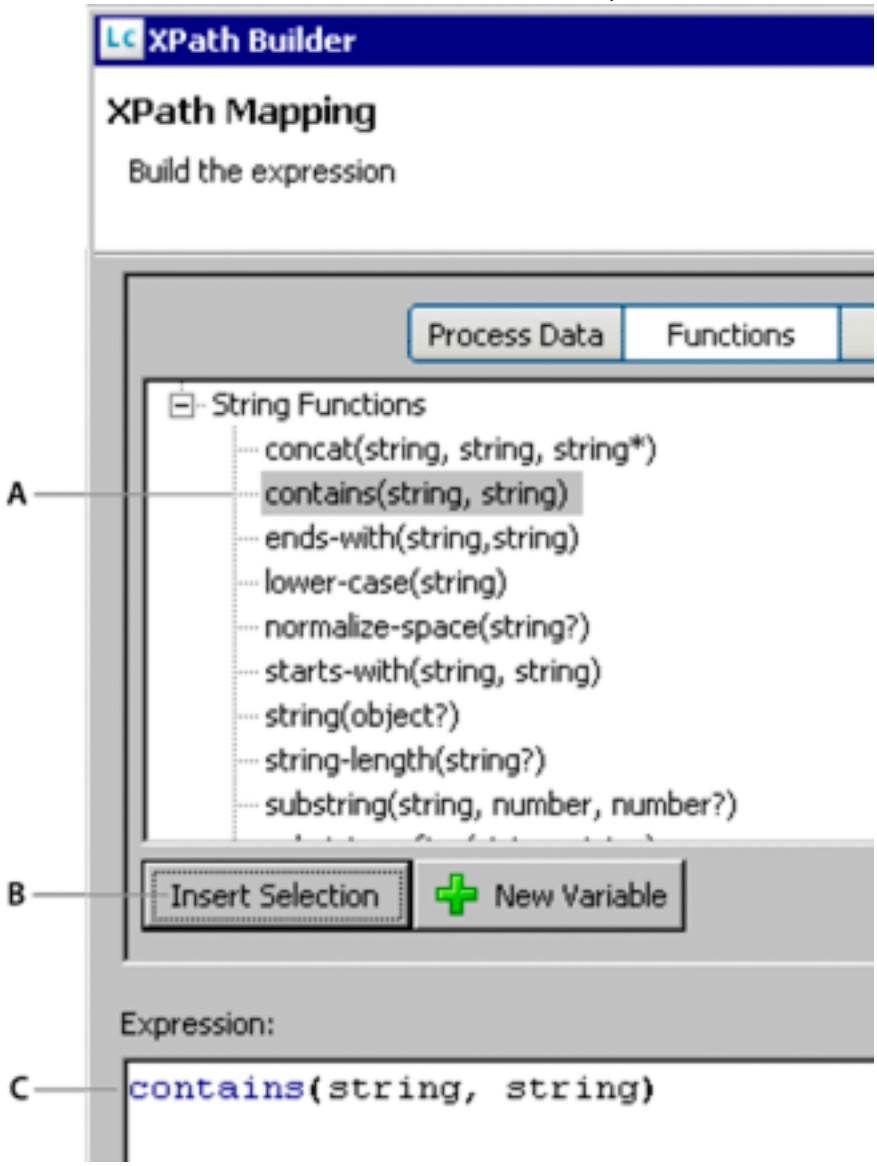
Workbench supports XPath expressions that include functions. Functions are evaluated at run time.

Most functions require input parameters on which they act. Use the procedure in this topic to add a function to the expression work area and specify the input parameters.

**To build an expression that uses functions:**

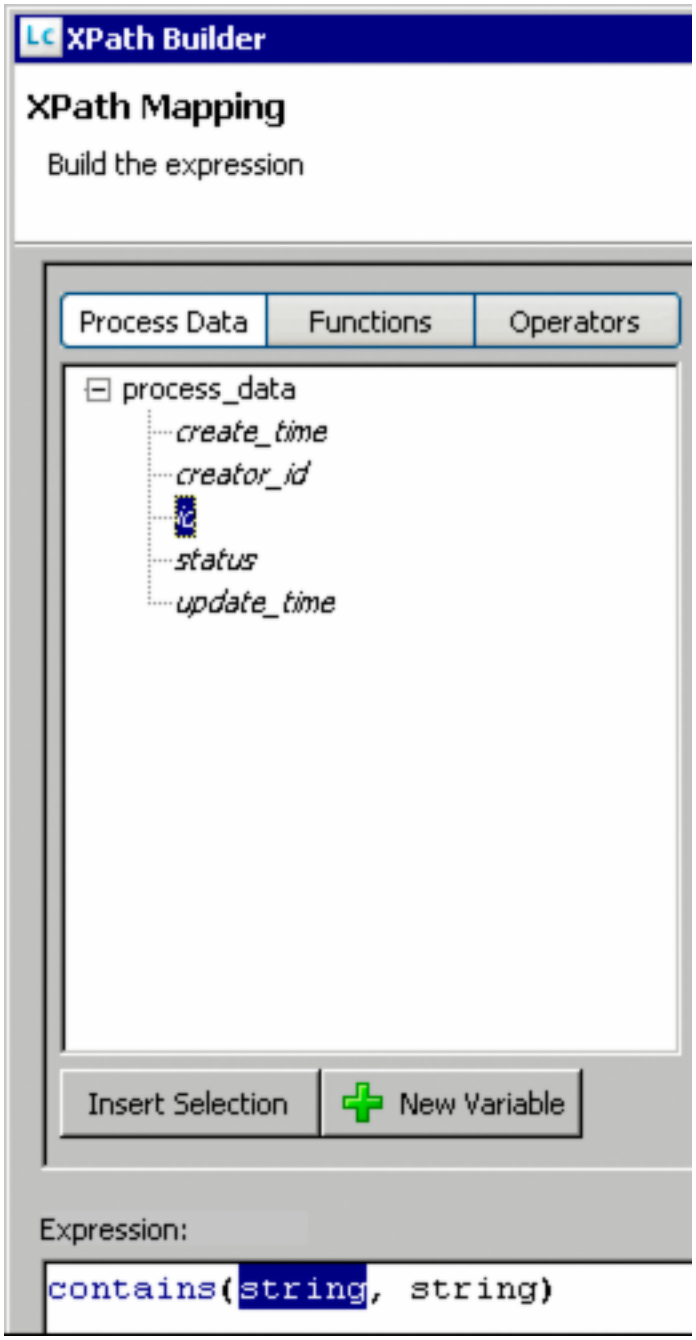
1) In XPath Builder, click Functions.
2) In the function library, expand function groups and locate the function that you want to use in the expression.
3) Choose either of these ways to add the function to the expression:
   – Select the function and click Insert Selection.
   – Double-click the function.

**A.** Selected function **B.** Click to insert **C.***Inserted expression*




*If the function requires input parameters, placeholder text for each parameter appears within parentheses after the function name. Parameters are separated by a comma. If the function requires no input parameters, no text appears within the parentheses. For example, the function* `contains(string, string)` *requires two* `string` *parameters.*

4) If the function that you added requires one or more input parameters, specify the values to use for each parameter:
   – In the expression work area, select the placeholder text for a parameter of the function. For example, for the function `contains(string, string)`, you would select one of the `string` parameters.




– While the function parameter is selected, enter the text to use for the parameter value:
  - To use the results of another function, select the function in the function library and click Insert Selection.
  - To use data from the data tree, click Process Data to display the process data tree, select the data item to use as the parameter and click Insert Selection.
  - To enter other text, type the value.

  *The value that you specified replaces the input parameter that you selected in the work area.*

5) If you used another function as an input parameter, repeat step 4 to specify the parameter values for the function.
6) When the expression is complete, click OK.

RELATED LINKS:

Editing text in the expression work area

Building simple XPath expressions

Building XPath expressions that use operators

Creating XPath expressions

XPath Builder (Process Properties)

XPath function reference

## Building XPath expressions that use operators

Workbench supports XPath expressions that include operators. Operators are evaluated at run time.

Almost all expressions that include operators require that a value is included on either side of the operation. For example, the less-than operator (<) requires two values to compare.

Only the unary minus expression uses an operator with only one value; unary minus expressions include a minus sign (-) followed by a value. The unary minus expression −*value* returns the negative of the value *value*.

**To build an expression that uses operators:**

1) Enter the value to use on the left side of the operator:
   – To use data from the data tree, click Process Data to view the data tree, select the data item to use as the value, and click Insert Selection.
   – To use the results of a function, click Functions to view the Functions library, select the function, and click Insert Selection.
   – To enter other text, type the value.
2) Click Operators to view the Operators library.
3) Locate the operator that you want to use and choose either of these ways to add it to the expression:
   – Select the operator and click Insert Selection.
   – Double-click the operator.

**A.** Selected operator **B.** Click to insert **C.***Inserted operator*

4) Enter the value to use on the right side of the operator:
   - To use data from the data tree, click Process Data to view the data tree, select the data item to use as the value, and click Insert Selection.
   - To use the results of a function, click Functions to view the Functions library, select the function, and click Insert Selection.
   - To enter other text, type the value.

5) When the expression is complete, click OK.

RELATED LINKS:

Editing text in the expression work area

Building simple XPath expressions

Building XPath expressions that use functions

Creating XPath expressions

XPath Builder (Process Properties)

XPath function reference

## Building XPath expressions that use Common Variables

Common variables are common, and available to all long-lived processes whereas process variables are only available to a specific process. Common variables can be utilized in a process for tracking purposes and to search for processes in Workspace.

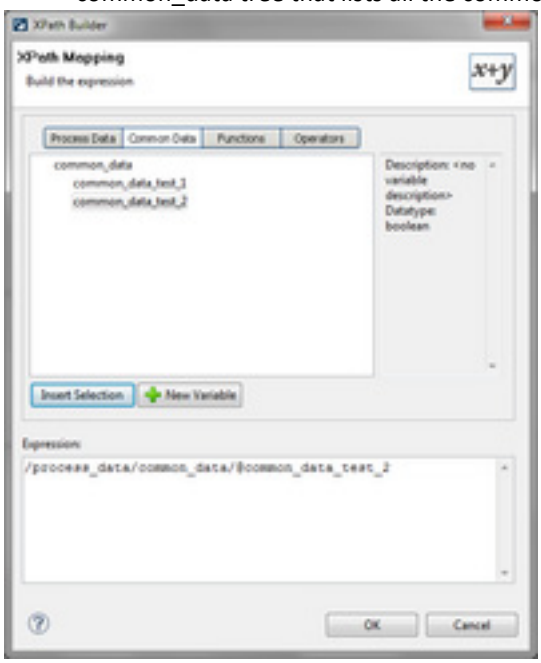
You can also build expressions with nodes of XML variables that conform to an XSD that references XSD files.

**To build an expression that uses common variables:**

1) Navigate the common data tree to locate the item that you want to include in the expression.
2) Choose either of these ways to add the item to the expression:
   – Select the item and click Insert Selection.

     **NOTE:** To build expressions that contain nodes of XML variables that conform to an XSD that references XSD files, drill down to the nodes as desired in the XPATH builder.
   – Double-click the item.
3) Repeat step 2 as required.
4) When the expression is complete, click OK.

common_data tree that lists all the common variables in the XPath Builder




RELATED LINKS:

Building simple XPath expressions

Building XPath expressions that use functions

Building XPath expressions that use operators

Creating XPath expressions

XPath Builder (Process Properties)

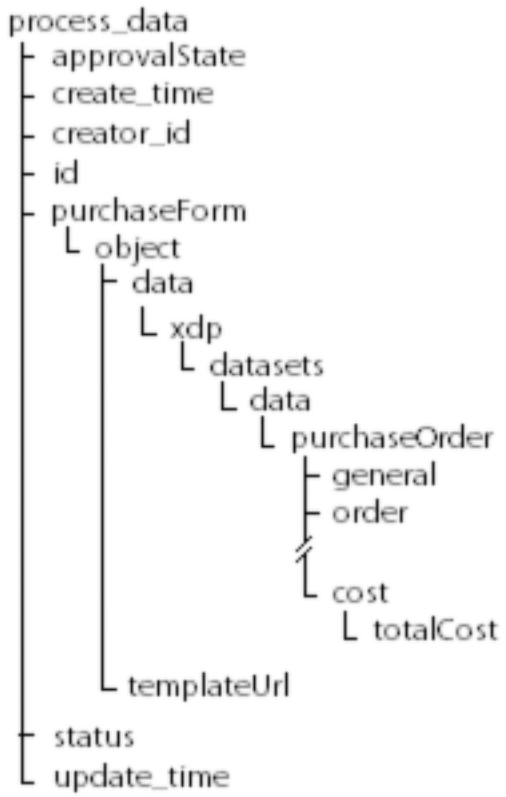
XPath function reference

# 15.4. Examples of XPath expressions

This section provides examples about applying XPath expressions to data from an example process data tree. The process implements a purchase order process. Routing decisions are made based on the

amount of the purchase. A form is used to collect data about the purchase and is routed to several people to approve the purchase and place the order.

The process includes a `string` variable named `approvalState` that tracks whether the purchase request has been approved. Form data is saved in an `XFAForm` variable named `purchaseForm`. The root node of the form schema is named `purchaseOrder`. The process data tree is illustrated in the following diagram.




The nodes below the `purchaseOrder` node represent the form schema. The `totalCost` node is bound to a field on the form that stores the total cost of the requested purchase.

RELATED LINKS:

Retrieving form field values

Changing values in form data

Retrieving node sets

Converting data types

## Retrieving form field values

The example process (see ExampleXPath expressions) dictates that purchases of a total cost less than $300 do not require a manager's approval. Those purchases can be routed to an administrator for immediate ordering. A route condition decides whether to route the purchases to an administrator or a manager. The condition uses an expression to access the value of the `totalCost` field on the form.

The process data tree is illustrated in the following diagram.

```
process_data
├ approvalState
├ create_time
├ creator_id
├ id
├ purchaseForm
│ └ object
│   ├ data
│   │ └ xdp
│   │   └ datasets
│   │     └ data
│   │       └ purchaseOrder
│   │         ├ general
│   │         ├ order
│   │         //
│   │         └ cost
│   │           └ totalCost
│   └ templateUrl
├ status
└ update_time
```

The following expression retrieves the value of the `totalCost` field on the form:

```
/process_data/purchaseForm/object/data/xdp/datasets/data/purchaseOrder/c
ost/totalCost
```

**NOTE:** When extracting the date from a form field and processing it using a Date or Date-Time function in an XPath expression, the extracted date must be in the format that is valid for the function. (See Dateand time parameters.)

## Changing values in form data

The example process (see ExampleXPath expressions) requires that the purchase amounts be specified in a different currency. You can use the execute operation of the Set Value service to change the values in the form fields.

The process data tree is illustrated in the following diagram.

```
process_data
├ approvalState
├ create_time
├ creator_id
├ id
├ purchaseForm
│ └ object
│   ├ data
│   │ └ xdp
│   │   └ datasets
│   │     └ data
│   │       └ purchaseOrder
│   │         ├ general
│   │         ├ order
│   │         ⋮
│   │         └ cost
│   │           └ totalCost
│   └ templateUrl
├ status
└ update_time
```

To perform mathematical operations on form field values, you first must change the field value to a number type. Changing the value of the `totalCost` field by an exchange rate of 0.82 requires the following expression for the Expression property of the service's execute operation:

```
number(/process_data/purchaseForm/object/data/xdp/datasets/data/purchase
Order/cost/totalCost) *0.82)
```

## Retrieving node sets

It is often useful to retrieve a group of nodes from the data model. For example, to use the renderForm operation of the Forms service, you need to specify the form data.

The process data tree is illustrated in the following diagram.

```
process_data
├ approvalState
├ create_time
├ creator_id
├ id
├ purchaseForm
│ └ object
│   ├ data
│   │ └ xdp
│   │   └ datasets
│   │     └ data
│   │       └ purchaseOrder
│   │         ├ general
│   │         ├ order
│   │         ⋮
│   │         └ cost
│   │           └ totalCost
│   └ templateUrl
├ status
└ update_time
```

You can use the following XPath expression to retrieve all the nodes below the `data` node:

```
/process_data/purchaseForm/object/data/*
```

The `*` symbol is the XPath syntax that represents all child nodes.

## Converting data types

You can change some data types to another data type. For example, you can convert your data to a different type so that you can use it as a property value for a service operation. The following XPath functions are available in Workbench for changing data types:

- The `number` function converts data in a node to a number.
- The `deserialize` function converts the string value in a node to an XML document
- The `serialize` function converts an XML document to a string value.

### Converting XML to string data

Use the `serialize` XPath function to convert an XML document to a `string` value. Because form data is represented using XML, you can use the `serialize` function to convert form data to `string` data.

### Converting to string data

Use the `string` XPath function to convert data types to string representations. This function is useful for converting data that you retrieve from a database or an XML file and pass it to an operation as string data.

### Converting string data to XML

Use the deserialize function to convert string data to an XML document. This function is useful when XML data is provided as a string, and you want to convert it to XML to use XPath to more easily work with the data.

The following XPath expression converts the process string variable named `strvar` to an XML document:

```
deserialize(/process_data/@strvar)
```

The XML document needs to be saved in an XML variable.

**TIP:** If you associate a schema with the XML variable, the XML is represented as nodes in the process data tree of XPath Builder. Without the schema, only the name of the XML variable appears.

RELATED LINKS:

Building expressions by using XPath Builder

XPath Builder (Process Properties)

XPath function reference

SetValue

## 15.5. Accessing data in data collections

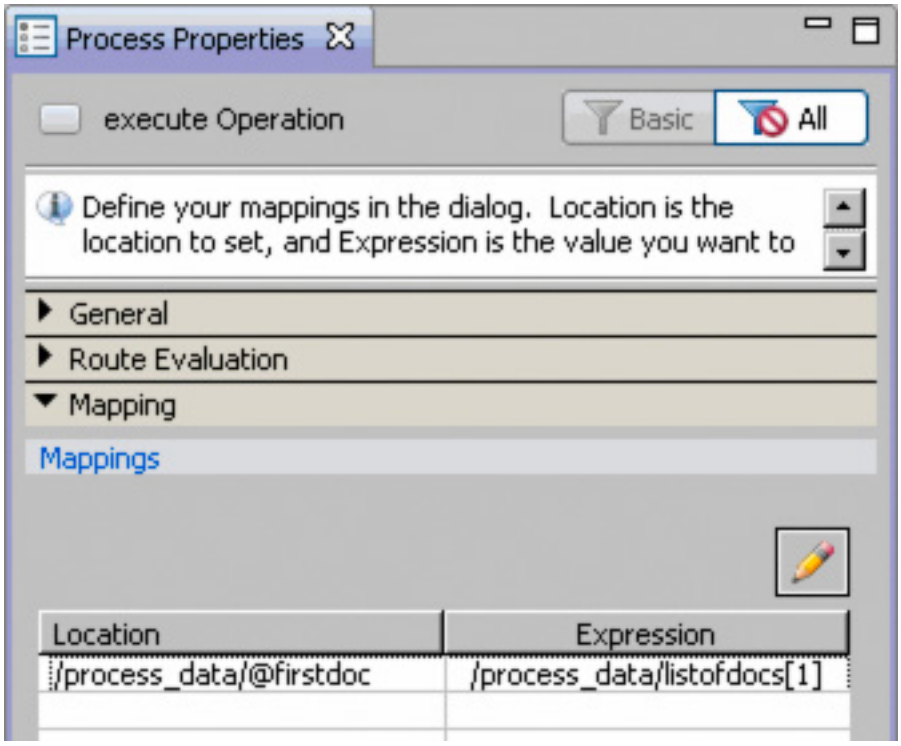
Process `list` and `map` variables are referenced in XPath expressions by name in the same way that other types of variables are referenced. However, to access specific data items in `list` and `map` variables, you also need to indicate which data item in a `list` or `map` variable you want to set or access.

### list variables

To reference an entire `list` variable, you use the name of the `list` variable. For example, to copy all of the data from a `list` variable named `original_list` to a `list` variable named `copy_list`, the XPath expression that you use to reference `original_list` is `/process_data/original_list`.

To indicate the data item in a `list` variable, you need to specify the index position of the item. For example, a process stores document data in a `list` variable named `listofdocs`. You can use the execute operation of the Set Value service to retrieve the first document in the list and save it in a `document` variable named `firstdoc`. The XPath expression that returns the first document in the list is `/process_data/listofdocs[1]`.

**NOTE:** Indexes for `list` variables are 1-based so that the first item in the list has an index of 1.

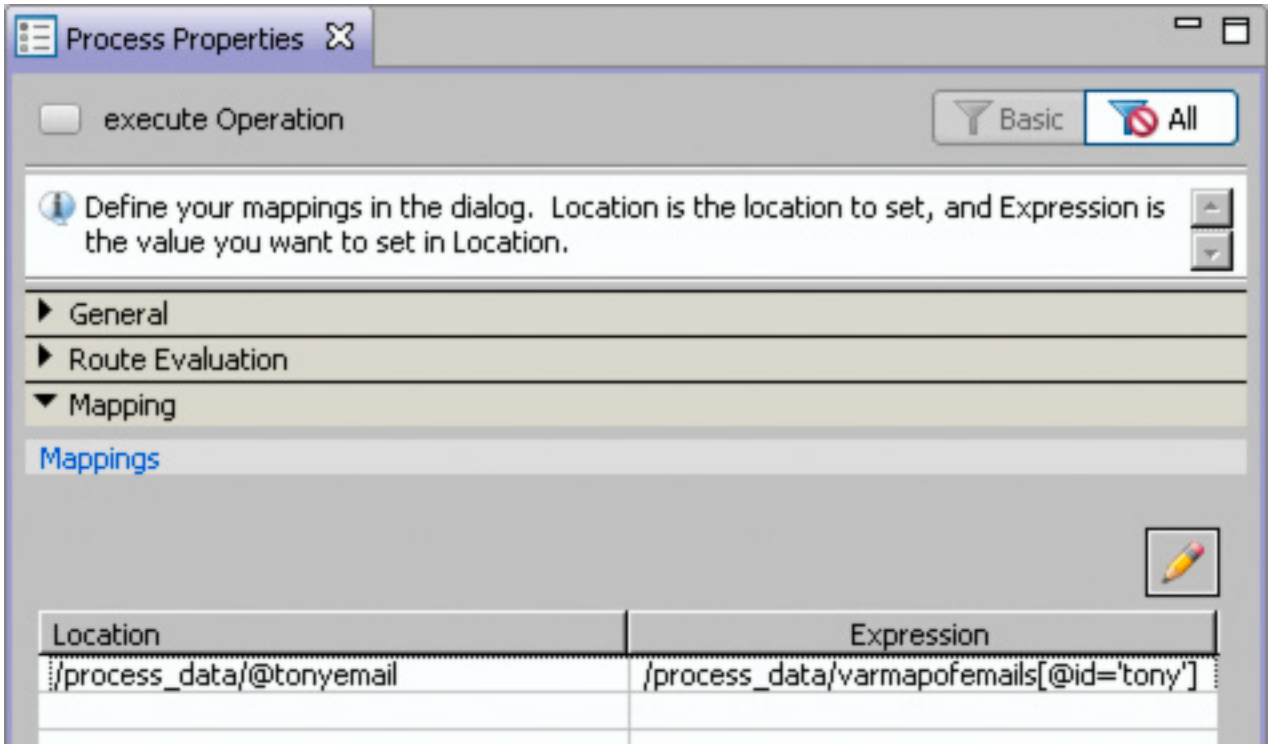



**NOTE:** When saving the results of service operations in `list` variables, the returned data items are appended to the list after any existing items. To replace the items in a list, you specify the index of the first item (for example, `/process_data/listVar[1]`).

## map variables

To reference an entire `map` variable, you use the name of the `map` variable. For example, to copy all of the data from a `map` variable named `original_map` to a `map` variable named `copy_map`, the XPath expression that you use to reference `original_map` is `/process_data/original_map`.

To indicate the data item in a `map` variable, you need to specify the key for the key-value pair. For example, a process stores names and email addresses in a `map` variable named `varmapofemails`. Names are used as the key, and the email addresses are the corresponding values. You can use the execute operation of the Set Value service to retrieve Tony's email address and store it in a `string` variable named `tonyemail`. The XPath expression that returns the value in the `map` variable that corresponds to the key of `tony` is `/process_data/mapofemails[@id='tony']`.




**NOTE:** Unlike `list` variables, the order in which data items are stored in `map` variables can be different each time you use the `map` variable. When referencing data items in `map` variables, you cannot make any assumptions about the order of the data items.

RELATED LINKS:

Using XPath expressions as list indexes and map keys

XPath Builder (Process Properties)

Building expressions by using XPath Builder

XPath function reference

Example XPath expressions

## 15.6. Using XPath expressions as list indexes and map keys

When referencing process `list` and `map` variables in XPath expressions, you can use other process variables for the list's `index` value or the map's `key` value.

For example, a `list` variable named `listvar` holds a collection of customer names. An `integer` variable named `intvar` holds the index of the data item in `listvar` that you want to retrieve. The following XPath expression returns the data item from `listvar`:

```
/process_data/listvar[number(/process_data/@intvar)]
```

**NOTE:** The `number` function ensures that the data in the `/process_data/@intvar` node is interpreted as a number. (See number.) If a `string` value is used as the list index, the expression returns the first item in the list.

A `map` variable named `mapvar` holds customer addresses and uses the customer name as the key. The customer name that you want the address for is stored in a `string` variable named `stringvar`. The following XPath expression retrieves the address from `mapvar`, which has, as its key, the customer name stored in `stringvar`:

```
/process_data/mapvar[@id='/process_data/@stringvar']
```

RELATED LINKS:

Accessing data in data collections

XPath Builder (Process Properties)

Building expressions by using XPath Builder

XPath function reference

Example XPath expressions

# 15.7. XPath function reference

This appendix provides information about the functions and operators that are available for use in XPath expressions in Workbench. The functions and operators are based on the XML Path Language version 1.0 W3C Recommendation. The recommendation is available at www.w3c.org/TR/xpath

.

The Workbench implementations may not function exactly as they are described in the W3C recommendation in all cases. Also not all the features implemented in the W3C recommendation are available.

For information about operators that are used in XPath functions, see Node set functions.

RELATED LINKS:

Special characters in function syntax

Date and time parameters

Time zone descriptive identifiers

## Special characters in function syntax

The syntax of some XPath functions include the question mark (?) character and the asterisk (*):

- A question mark that follows a function parameter indicates that the parameter is optional. For example, the `object` parameter of the `number` function is optional: `number(object?)`.
- An asterisk that follows a function parameter indicates that the function can take zero or more values for the parameter. For example, zero or more string values can be provided as the value for the third `string` parameter in the `concat` function,: `concat(string, string, string*)`.

## Boolean functions

This section describes the Boolean functions that are available in expressions.

### boolean

Converts a given value to an equivalent `boolean` value. This function is useful for testing for the existence of a node in the data model.

*Syntax*

```
boolean(Expression)
```

*Parameters*

```
Expression
```

*Returns*

A `boolean` value.

*Example*

The following XPath expression returns true because the `stringVar` node exists in the process data model:

```
boolean(/process_data/@stringVar)
```

### false

Returns `false`. This function is useful when you need to compare boolean values to a value of `false`, or set the value of a data item to `false`.

*Syntax*

```
false()
```

*Parameters*

None.

*Returns*

A `boolean` value of `false`.

*Example*

The following route condition tests whether the `/process_data/stringVar node` does not exist in the process data model:

```
/process_data/@stringVar = false()
```

The following Set Value mapping sets the value of the `booleanVar` node to `false`:

```
/process_data/@booleanVar = false()
```

### not

The inverse value of a given `boolean` value.

*Syntax*

```
not(boolean)
```

*Parameters*

A `boolean` value.

*Returns*

A `boolean` value of true if the parameter `boolean` is false, otherwise false.

**true**

Returns `true`. This function is useful when you must compare boolean values to a value of `true`, or set the value of a data item to `true`.

*Syntax*

```
true()
```

*Parameters*

None.

*Returns*

A `boolean` value of `true`.

*Example*

The following route condition tests whether the `/process_data/stringVar node` exists in the process data model:

```
/process_data/@stringVar = true()
```

The following Set Value mapping sets the value of the `booleanVar` node to `true`:

```
/process_data/@booleanVar = true()
```

## Collection functions

This section describes the collection functions that are available for use on `list` and `map` values in expressions.

**empty-list**

Returns an empty `list` value. This function is useful when you want to clear the values of an existing list value. For example, to remove all the items in a list value, use the Set Value service to set the value of the list to `empty-list()`.

*Syntax*

```
empty-list()
```

*Parameters*

None.

*Returns*

A `list` value that contains no items.

*Example*

The following table represents a mapping in an execute operation that the SetValue service provides. The mapping uses the `empty-list` function to remove the items from the `listVar` variable, which holds `document` values.

| Location | Expression |
|---|---|
| `/process_data/listVar` | `empty-list()` |

After the list values are cleared, the `listVar` variable is still configured to hold `document` values.

**empty-map**

Returns an empty `map` value. This function is useful when you want to clear the values of an existing map value. For example, to remove all the items in a map value, use the Set Value service to set the value of the map to `empty-map()`.

*Syntax*

```
empty-map()
```

*Parameters*

None.

*Returns*

A `map` value that contains no items.

*Example*

The following table represents a mapping in an execute operation that the SetValue service provides. The mapping uses the `empty-map` function to remove the items from the `mapVar` variable, which holds `string` values.

| Location | Expression |
|---|---|
| `/process_data/mapVar` | `empty-map()` |

After the map values are cleared, the `mapVar` variable is still configured to hold `string` values.

### get-map-keys

Retrieves the keys from a `map` value.

*Syntax*

```
get-map-keys(Expression)
```

*Parameters*

`Expression` is an XPath expression that resolves to a `map` value.

*Returns*

A `list` value that contains `string` values that represent `map` keys.

*Example*

A `map` variable named `varmap` includes the key-value pairs (`username, admin`) and (`password, 1234`). The following example returns a `list` value that includes the `string` values `username` and `password`:

```
get-map-keys(/process_data/varmap)
```

### get-collection-size

Retrieves the number of items in a data collection.

*Syntax*

```
get-collection-size(node-set)
```

*Parameters*

`node-set` is an XPath expression that resolves to the data node that represents a `list` or `map` variable.

*Returns*

An `integer` that contains the number of data items in the `node-set`.

*Example*

A `list` variable named `varlist` contains six data items. The following example returns `6`:

```
get-collection-size(/process_data/varlist)
```

### get-list-item-count

Returns the number of items in a list that are equal to a given string value. Optionally, returns the number of items that have a property that is equal to a given string value.

*Syntax*

```
get-list-item-count(list, strToMatch [, strPropertyName])
```

*Parameters*

*list* is an XPath expression that resolves to the list value.

*strToMatch* is the value to match with items in the list. This value can be a string value or any type of value that can be coerced to a string, such as a number or XML.

*strPropertyName* is an optional string value that represents the name of the property of the items in the list to match with *strToMatch*.

*Returns*

An `int` value that represents the number of matching items in the list.

*Example*

The list variable named *strList* contains the items 'blue','red', and 'blue'. The following expression returns the value 2:

```
get-list-item-count(/process_data/strList, 'blue')
```

The list variable named docList contains the attachments and notes that are submitted with a task from Workspace. The list contains two file attachments and one note. The following expression returns the value 1:

```
get-list-item-count(/process_data/docList, 'note', 'wsattachtype')
```

For an example that uses this function with a TaskResultCollection value, see Assessing review and approval results.

**get-list-item-percentage**

Returns the percentage of items in a list that are equal to a given string value. Optionally, returns the number of items that have a property that is equal to a given string value.

*Syntax*

```
get-list-item-percentage(list, strToMatch [, strPropertyName])
```

*Parameters*

*list* is an XPath expression that resolves to the list value.

*strToMatch* is the string value to match with items in the list. This value can be any type that can be coerced to a string, such as a number or XML.

*strPropertyName* is an optional string value that represents the name of the property of the items in the list to match with *strToMatch*.

*Returns*

A double precision value that represents the number of matching items in the list.

**TIP:** If you save the returned value in a string variable, the value is coerced to a string.

*Example*

The list variable named strList contains the items 'blue', 'red', and 'blue'. The following expression returns the value 66.667:

```
get-list-item-percentage(/process_data/strList, 'blue')
```

The list variable named docList contains the attachments and notes that are submitted with a task from Workspace. The list contains two file attachments and one note. The following expression returns the value 33.333:

```
get-list-item-percentage(/process_data/docList, 'note', 'wsattachtype')
```

**get-map-values**

The get-map-values function returns the values that are stored in a `map` variable.

*Syntax*

```
get-map-values(Expression)
```

*Parameters*

*Expression* is an XPath expression that resolves to a map value from which you can retrieve the keys.

*Returns*

A `list` value that contains `string` values that represent the `map` values.

*Example*

A `map` variable named `varmap` includes the key-value pairs (`username`, `admin`) and (`password`, `1234`). The following expression returns a `list` value that includes the `string` values `admin` and `1234`:

```
get-map-values(/process_data/varmap)
```

**get-map-values-for-keys**

Returns the values stored in a `map` value that correspond with a given list of keys.

*Syntax*

```
get-map-values-for-keys(Expression1, Expression2)
```

*Parameters*

*Expression1* is an XPath expression that resolves to a `map` value from which you can retrieve the values.

*Expression2* is an XPath expression that resolves to a `list` value that contains the keys for the values that you want to retrieve.

*Returns*

A `list` value that contains `string` values that represent the `map` values.

*Example*

A `map` variable named `varmap` stores customer email addresses. A customer ID is used as the key, and the email address is the value. A `list` variable named `varlist` contains several customer IDs. The following expression returns the values from `varmap` that correspond with the customer IDs stored in `varlist`:

```
get-map-values-for-keys(/process_data/varmap, /process_data/varlist)
```

## Date-Time functions

This section describes the date-time functions that are available in expressions.

### add-days-to-date

Adds the specified number of days to a given date.

*Syntax*

```
add-days-to-date(strDate1, days)
```

*Parameters*

`strDate1` is a `string` that represents the date and is specified in the format `yyyy-mm-dd`. For more information about date and time parameters, see Date andtime parameters.

`days` is a `number` that holds the number of days to add to the given date. A negative number for `days` subtracts days from the given date.

*Returns*

A `string` value in the format `yyyy-mm-dd`.

*Example*

The following expression returns a `string` of value `2001-10-30`:

```
add-days-to-date("1999-11-28", 337)
```

### add-days-to-dateTime

Adds the specified number of days to a given date and time.

*Syntax*

```
add-days-to-dateTime(strDateTime1, days)
```

*Parameters*

`strDateTime1` is a `string` that represents the date and time and is specified in the format `yyyy-mm-ddThh:mm:ssZ`. For more information about date and time parameters, see Dateand time parameters.

`days` is a `number` that holds the number of days to add to the given date and time. A negative number for `days` subtracts days from the given date and time.

*Returns*

A `string` value in the format `yyyy-mm-ddThh:mm:ssZ`.

*Example*

The following expression returns a `dateTime` value of `2000-10-30T11:12:00Z`:

```
add-days-to-dateTime("1999-11-28T11:12:00Z",337)
```

**add-hours-to-dateTime**

Adds the specified number of hours to a given date and time.

*Syntax*

```
add-hours-to-dateTime(strDateTime1, hours)
```

*Parameters*

`strDateTime1` is a `string` that represents the date and time and is specified in the format `yyyy-mm-ddThh:mm:ssZ`. For more information about date and time parameters, see Dateand time parameters.

`hours` is a `number` that holds the number of hours to add to the given date and time. A negative number for `hours` subtracts hours from the given date and time.

*Returns*

A `string` value in the format `yyyy-mm-ddThh:mm:ssZ`.

*Example*

The following expression returns a `string` value of `1999-11-28T03:12:00Z`:

```
add-hours-to-dateTime("1999-11-28T11:12:00Z",-8)
```

**add-minutes-to-dateTime**

Adds the specified number of minutes to a given date and time.

*Syntax*

```
add-minutes-to-dateTime(strDateTime1, minutes)
```

*Parameters*

`strDateTime1` is a `string` that represents the date and time and is specified in the format `yyyy-mm-ddThh:mm:ssZ`. For more information about date and time parameters, see Dateand time parameters.

`minutes` is a `number` that holds the number of minutes to add to the given date and time. A negative number for `minutes` subtracts minutes from the given date and time.

*Returns*

A `string` value in the format `yyyy-mm-ddThh:mm:ssZ`.

*Example*

The following expression returns a `string` value of `1999-11-28T11:22:00Z`:

```
add-minutes-to-dateTime("1999-11-28T11:12:00Z",10)
```

**add-months-to-date**

Adds the specified number of months to a given date.

*Syntax*

```
add-months-to-date(strDate1, months)
```

*Parameters*

`strDate1` is a `string` that represents the date and is specified in the format `yyyy-mm-dd`. For more information about date and time parameters, see Date andtime parameters.

`months` is a `number` that holds the number of months to add to the given date. A negative number for `months` subtracts months from the given date.

*Returns*

A `string` value in the format `yyyy-mm-dd`.

*Example*

The following expression returns a `string` value of `2000-10-28`:

```
add-months-to-date('1999-11-28',11)
```

**add-months-to-dateTime**

Adds the specified number of months to a given date and time.

*Syntax*

```
add-months-to-dateTime(strDateTime1, months)
```

*Parameters*

`strDateTime1` is a `string` that represents the date and time, and is specified in the format `yyyy-mm-ddThh:mm:ssZ`. For more information about date and time parameters, see Dateand time parameters.

`months` is a `number` that holds the number of months to add to the given date and time. A negative number for `months` subtracts months from the given date and time.

*Returns*

A `string` value in the format `yyyy-mm-ddThh:mm:ssZ`.

*Example*

The following expression returns a `string` value of `2000-10-28T11:12:00Z`:

```
add-months-to-dateTime('1999-11-28T11:12:00Z',11)
```

**add-seconds-to-dateTime**

Adds the specified number of seconds to a given date and time.

*Syntax*

```
add-seconds-to-dateTime(strDateTime1, seconds)
```

*Parameters*

`strDateTime1` is a `string` that represents the date and time and is specified in the format `yyyy-mm-ddThh:mm:ssZ`. For more information about date and time parameters, see Dateand time parameters.

`seconds` is a `number` that holds the number of seconds to add to the given date and time. A negative number for `seconds` subtracts seconds from the given date and time.

*Returns*

A `string` value in the format `yyyy-mm-ddThh:mm:ssZ`.

*Example*

The following expression returns a `string` value of 1999-11-28T11:11:30Z:

```
add-seconds-to-dateTime('1999-11-28T11:12:00Z',-30)
```

**add-years-to-date**

Adds the specified number of years to a given date.

*Syntax*

```
add-years-to-date(strDate1, years)
```

*Parameters*

`strDate1` is a `string` that represents the date and is specified in the format `yyyy-mm-dd`. For more information about date and time parameters, see Date andtime parameters.

`years` is a `number` that holds the number of years to add to the given date and time. A negative number for `years` subtracts years from the given date.

*Returns*

A `string` value in the format `yyyy-mm-ddThh:mm:ssZ`.

*Example*

The following expression returns a `string` value of `2001-11-28`:

```
add-years-to-date("1999-11-28",2)
```

**add-years-to-dateTime**

Adds the specified number of years to a given date and time.

*Syntax*

```
add-years-to-dateTime(strDateTime1, years)
```

*Parameters*

`strDateTime1` is a `string` that represents the date and time and is specified in the format `yyyy-mm-ddThh:mm:ssZ`. For more information about date and time parameters, see Dateand time parameters.

`years` is a `number` that holds the number of years to add to the given date and time. A negative number for `years` subtracts years from the given date and time.

*Returns*

A `string` value in the format `yyyy-mm-ddThh:mm:ssZ`.

*Example*

The following expression returns a `string` value of `2001-11-28T11:12:00Z`:

```
add-years-to-dateTime("1999-11-28T11:12:00Z",2)
```

**current-date**

Returns the current date.

*Syntax*

```
current-date()
```

*Parameters*

None.

*Returns*

A `string` value in the format `yyyy-mm-dd`.

**current-dateTime**

Returns the current date and time.

*Syntax*

```
current-dateTime()
```

*Parameters*

None.

*Returns*

A `string` value in the format `yyyy-mm-ddThh:mm:ssZ`.

**current-time**

Returns the current time.

*Syntax*

```
current-time()
```

*Parameters*

None.

*Returns*

A `string` value in the format `hh:mm:ss`.

**date-equal**

Compares two given dates to see if they are equal.

*Syntax*

```
date-equal(strDate1, strDate2)
```

*Parameters*

`strDate1` is a `string` that represents one date to compare and is specified in the format `yyyy-mm-dd`.

`strDate2` is a `string` that represents the other date to compare and is specified in the format `yyyy-mm-dd`.

For more information about date and time parameters, see Dateand time parameters.

*Returns*

A Boolean value of `true` if the date that `strDate1` represents is equal to the date that `strDate2` represents, otherwise `false`.

**date-greater-than**

Compares two given dates to see if one is greater than the other.

*Syntax*

```
date-greater-than(strDate1, strDate2)
```

*Parameters*

`strDate1` is a `string` that represents one date to compare and is specified in the format `yyyy-mm-dd`.

`strDate2` is a `string` that represents the other date to compare and is specified in the format `yyyy-mm-dd`.

For more information about date and time parameters, see Dateand time parameters.

*Returns*

A Boolean value of `true` if the date that `strDate1` represents is greater than the date that `strDate2` represents, otherwise `false`.

**date-less-than**

Compares two given dates to see if one is less than the other.

*Syntax*

```
date-less-than(strDate1, strDate2)
```

*Parameters*

`strDate1` is a `string` that represents one date to compare and is specified in the format `yyyy-mm-dd`.

`strDate2` is a `string` that represents the other date to compare and is specified in the format `yyyy-mm-dd`.

For more information about date and time parameters, see Dateand time parameters.

*Returns*

A Boolean value of `true` if the date that `strDate1` represents is less than the date that `strDate2` represents, otherwise `false`.

**dateTime-equal**

Checks whether two given dates and times are equal.

*Syntax*

```
dateTime-equal(strDateTime1, strDateTime2)
```

*Parameters*

`strDateTime1` is a `string` that represents one date and time to compare and is specified in the format `yyyy-mm-ddThh:mm:ssZ`.

`strDateTime2` is a `string` that represents the other date and time to compare and is specified in the format `yyyy-mm-ddThh:mm:ssZ`.

For more information about date and time parameters, see Dateand time parameters.

*Returns*

A Boolean value of `true` if the date and time that `strDateTime1` represents is equal to the date and time that `strDateTime2` represents, otherwise `false`.

**dateTime-greater-than**

Checks whether a given date and time is greater than another given date and time.

*Syntax*

```
dateTime-greater-than(strDateTime1, strDateTime2)
```

*Parameters*

`strDateTime1` is a `string` that represents one date and time to compare and is specified in the format `yyyy-mm-ddThh:mm:ssZ`.

`strDateTime2` is a `string` that represents the other date and time to compare and is specified in the format `yyyy-mm-ddThh:mm:ssZ`.

For more information about date and time parameters, see Dateand time parameters.

*Returns*

A Boolean value of `true` if the date and time that `strDateTime1` represents is greater than the date and time that `strDateTime2` represents, otherwise `false`.

### dateTime-less-than

Checks whether a given date and time is less than another given date and time.

*Syntax*

```
dateTime-less-than(strDateTime1, strDateTime2)
```

*Parameters*

`strDateTime1` is a `string` that represents one date and time to compare and is specified in the format `yyyy-mm-ddThh:mm:ssZ`.

`strDateTime2` is a `string` that represents the other date and time to compare and is specified in the format `yyyy-mm-ddThh:mm:ssZ`.

For more information about date and time parameters, see Dateand time parameters.

*Returns*

A Boolean value of `true` if the date and time that `strDateTime1` represents is less than the date and time that `strDateTime2` represents, otherwise `false`.

### get-day-from-date

Returns the day part of a given date.

*Syntax*

```
get-day-from-date(strDate)
```

*Parameters*

`strDate` is a `string` that represents the date for which the day is returned and is specified in the format `yyyy-mm-dd`. For more information about date and time parameters, see Dateand time parameters.

*Returns*

A `number` that holds the day part of the given date.

### get-day-from-dateTime

Returns the day part of a given date and time.

*Syntax*

```
get-day-from-dateTime(strDateTime)
```

*Parameters*

`strDateTime` is a `string` that represents the date for which the day is returned and is specified in the format `yyyy-mm-ddThh:mm:ssZ`. For more information about date and time parameters, see Dateand time parameters.

*Returns*

A `number` that holds the day part of the given date and time.

**get-days-from-date-difference**

Calculates the difference between the day part of two given dates and returns the difference.

*Syntax*

```
get-days-from-date-difference(strDate1, strDate2)
```

*Parameters*

`strDate1` is a `string` that represents one date to compare and is specified in the format `yyyy-mm-dd`.

`strDate2` is a `string` that represents the other date to compare and is specified in the format `yyyy-mm-dd`.

For more information about date and time parameters, see Dateand time parameters.

*Returns*

A `number` that holds the difference between the day part of the given dates.

*Example*

The following expression returns a `number` of value `337`:

```
get-days-from-date-difference('2000-10-30','1999-11-28')
```

**get-days-from-dateTime-difference**

Calculates the difference between the day part of two given date and time values and returns the difference.

*Syntax*

```
get-days-from-dateTime-difference(strDateTime1,strDateTime2)
```

*Parameters*

`strDateTime1` is a `string` that represents one date and time to compare and is specified in the format `yyyy-mm-ddThh:mm:ssZ`.

`strDateTime2` is a `string` that represents the other date and time to compare and is specified in the format `yyyy-mm-ddThh:mm:ssZ`.

For more information about date and time parameters, see Dateand time parameters.

*Returns*

A `number` that holds the difference between the day part of the given date and time values.

*Example*

The following expression returns a `number` of value `337`:

```
get-days-from-date-Time-difference('2000-10-30T11:12:20Z','1999-11-28T09
:00:05Z')
```

**get-hours-from-dateTime**

Returns the hours part of a given date and time.

*Syntax*

```
get-hours-from-dateTime(strDateTime)
```

*Parameters*

`strDateTime` is a `string` that represents the date and time for which the hours is returned, and is specified in the format `yyyy-mm-ddThh:mm:ssZ`. For more information about date and time parameters, see Dateand time parameters.

*Returns*

A `number` that holds the hours part of the given date and time.

**get-hours-from-dateTime-difference**

Calculates the difference between the hour part of two given date and time values and returns the difference.

*Syntax*

```
get-hours-from-dateTime-difference(strDateTime1,strDateTime2)
```

*Parameters*

`strDateTime1` is a `string` that represents one date and time to compare and is specified in the format `yyyy-mm-ddThh:mm:ssZ`.

`strDateTime2` is a `string` that represents the other date and time to compare and is specified in the format `yyyy-mm-ddThh:mm:ssZ`.

For more information about date and time parameters, see Dateand time parameters.

*Returns*

A `number` that holds the difference between the hour part of the given date and time values.

*Example*

The following expression returns a `number` of value `2`:

```
get-hours-from-dateTime-difference('2000-10-30T11:12:20Z','1999-11-28T09
:00:05Z')
```

**get-hours-from-time**

Returns the hours part of a given time.

*Syntax*

```
get-hours-from-time(strTime)
```

*Parameters*

`strTime` is a `string` that represents the time and is specified in the format `hh:mm:ss`. For more information about date and time parameters, see Dateand time parameters.

*Returns*

A `number` that holds the hour part of the given time.

**get-minutes-from-dateTime**

Returns the minutes part of a given date and time value.

*Syntax*

```
get-minutes-from-dateTime(strDateTime)
```

*Parameters*

`strDateTime` is a `string` that represents the date and time and is specified in the format `yyyy-mm-ddThh:mm:ssZ`. For more information about date and time parameters, see Dateand time parameters.

*Returns*

A `number` that holds the minutes part of the given date and time value.

**get-minutes-from-dateTime-difference**

Calculates the difference between the minutes part of two given date and time values and returns the difference.

*Syntax*

```
get-minutes-from-dateTime-difference(strDateTime1,strDateTime2)
```

*Parameters*

`strDateTime1` is a `string` that represents one date and time to compare and is specified in the format `yyyy-mm-ddThh:mm:ssZ`.

`strDateTime2` is a `string` that represents the other date and time to compare and is specified in the format `yyyy-mm-ddThh:mm:ssZ`.

For more information about date and time parameters, see Dateand time parameters.

*Returns*

A `number` that holds the difference between the minutes part of the given date and time values.

*Example*

The following expression returns a `number` of value `12`:

```
get-minutes-from-dateTime-difference('2000-10-30T11:12:20Z','1999-11-28T
09:00:05Z')
```

**get-minutes-from-time**

Returns the minutes part of a given time.

*Syntax*

```
get-minutes-from-time(strTime)
```

*Parameters*

`strTime` is a `string` that represents the time and is specified in the format `hh:mm:ss`. For more information about date and time parameters, see Dateand time parameters.

*Returns*

A `number` that holds the minutes part of the given time.

**get-month-from-date**

Returns the month part of a given date.

*Syntax*

```
get-month-from-dateTime(strDate)
```

*Parameters*

`strDateTime` is a `string` that represents the date and is specified in the format `yyyy-mm-dd`. For more information about date and time parameters, see Dateand time parameters.

*Returns*

A `number` that holds the months part of the given date.

### get-month-from-dateTime

Returns the month part of a given date and time value.

*Syntax*

```
get-month-from-dateTime(strDateTime)
```

*Parameters*

`strDateTime` is a `string` that represents the date and time and is specified in the format `yyyy-mm-ddThh:mm:ssZ`. For more information about date and time parameters, see Dateand time parameters.

*Returns*

A `number` that holds the months part of the given date and time value.

### get-months-from-date-difference

Calculates the difference between the months part of two given date and time values and returns the difference.

*Syntax*

```
get-months-from-date-difference(strDate1,strDate2)
```

*Parameters*

`strDate1` is a `string` that represents one date to compare and is specified in the format `yyyy-mm-dd`.

`strDate2` is a `string` that represents the other date to compare and is specified in the format `yyyy-mm-dd`.

For more information about date and time parameters, see Dateand time parameters.

*Returns*

A `number` that holds the difference between the months part of the given date and time values.

*Example*

The following expression returns a `number` of value `11`:

```
get-months-from-date-difference('2000-10-30','1999-11-28')
```

### get-months-from-dateTime-difference

Calculates the difference between the months part of two given date and time values and returns the difference.

*Syntax*

```
get-months-from-dateTime-difference(strDateTime1,strDateTime2)
```

*Parameters*

`strDateTime1` is a `string` that represents one date and time to compare and is specified in the format `yyyy-mm-ddThh:mm:ssZ`.

`strDateTime2` is a `string` that represents the other date and time to compare and is specified in the format `yyyy-mm-ddThh:mm:ssZ`.

For more information about date and time parameters, see Dateand time parameters.

*Returns*

A `number` that holds the difference between the months part of the given date and time values.

*Example*

The following expression returns a `number` of value `11`:

```
get-months-from-dateTime-difference('2000-10-30T11:12:20Z','1999-11-28T0
9:00:05Z')
```

**get-seconds-from-dateTime**

Returns the seconds part of a given date and time value.

*Syntax*

```
get-seconds-from-dateTime(strDateTime)
```

*Parameter*

`strDateTime` is a `string` that represents the date and time and is specified in the format `yyyy-mm-ddThh:mm:ssZ`. For more information about date and time parameters, see Dateand time parameters.

*Returns*

A `number` that holds the seconds part of the given date and time value.

**get-seconds-from-dateTime-difference**

Calculates the difference between the seconds part of two given date and time values and returns the difference.

*Syntax*

```
get-seconds-from-dateTime-difference(strDateTime1,strDateTime2)
```

*Parameters*

`strDateTime1` is a `string` that represents one date and time to compare and is specified in the format `yyyy-mm-ddThh:mm:ssZ`.

`strDateTime2` is a `string` that represents the other date and time to compare and is specified in the format `yyyy-mm-ddThh:mm:ssZ`.

For more information about date and time parameters, see Dateand time parameters.

*Returns*

A `number` that holds the difference between the seconds part of the given date and time values.

*Example*

The following expression returns a `number` of value `15`:

```
get-seconds-from-dateTime-difference('2000-10-30T11:12:20Z','1999-11-28T
09:00:05Z')
```

**get-seconds-from-time**

Returns the seconds part of a given time.

*Syntax*

```
get-seconds-from-time(strTime)
```

*Parameters*

`strTime` is a `string` that represents the time, and is specified in the format `hh:mm:ss`. For more information about date and time parameters, see Dateand time parameters.

*Returns*

A `number` that holds the seconds part of the given time.

**get-timezone-from-date**

Returns the time zone part of a given date.

*Syntax*

```
get-timezone-from-date(strDate)
```

*Parameters*

`strDate` is a `string` that represents the date, and is specified in the format `yyyy-mm-dd`. For more information about date and time parameters, see Date andtime parameters.

*Returns*

A `number` that holds the time zone part of the given date.

### get-timezone-from-dateTime

Returns the time zone part of a given date and time value.

*Syntax*

```
get-timezone-from-dateTime(strDateTime)
```

*Parameters*

`strDateTime` is a `string` that represents the date and time, and is specified in the format `yyyy-mm-ddThh:mm:ssZ`. For more information about date and time parameters, see Dateand time parameters.

*Returns*

A `number` that holds the time zone part of the given date and time value.

### get-timezone-from-time

Returns the time zone part of a given time.

*Syntax*

```
get-timezone-from-time(strTime)
```

*Parameters*

`strTime` is a `string` that represents the time and is specified in the format `hh:mm:ss`. For more information about date and time parameters, see Dateand time parameters.

*Returns*

A `number` that holds the time zone part of the given time.

### get-year-from-date

Returns the year part of a given date.

*Syntax*

```
get-year-from-date(strDate)
```

*Parameters*

`strDate` is a `string` that represents the date and is specified in the format `yyyy-mm-dd`. For more information about date and time parameters, see Date andtime parameters.

*Returns*

A `number` that holds the year part of the given date.

### get-year-from-dateTime

Returns the year part of a given date and time value.

*Syntax*

```
get-year-from-dateTime(strDateTime)
```

*Parameters*

`strDateTime` is a `string` that represents the date and time and is specified in the format `yyyy-mm-ddThh:mm:ssZ`. For more information about date and time parameters, see Dateand time parameters.

*Returns*

A `number` that holds the year part of the given date and time value.

### get-years-from-dateTime-difference

Calculates the difference between the year part of two given date and time values and returns the difference.

*Syntax*

```
get-years-from-dateTime-difference(strDateTime1,strDateTime2)
```

*Parameters*

`strDateTime1` is a `string` that represents one date and time to compare and is specified in the format `yyyy-mm-ddThh:mm:ssZ`.

`strDateTime2` is a `string` that represents the other date and time to compare and is specified in the format `yyyy-mm-ddThh:mm:ssZ`.

For more information about date and time parameters, see Dateand time parameters.

*Returns*

A `number` that holds the difference between the year part of the given date and time values.

*Example*

The following expression returns a `number` of value `0`:

```
get-years-from-dateTime-difference('2000-10-30T11:12:20Z','1999-11-28T09
:00:05Z')
```

### get-years-from-date-difference

Calculates the difference between the year part of two given dates and returns the difference.

*Syntax*

```
get-years-from-date-difference(strDate1,strDate2)
```

*Parameters*

`strDate1` is a `string` that represents one date to compare and is specified in the format `yyyy-mm-dd`.

`strDate2` is a `string` that represents the other date to compare and is specified in the format `yyyy-mm-dd`.

For more information about date and time parameters, see Dateand time parameters.

*Returns*

A `number` that holds the difference between the year part of the given dates.

*Example*

The following expression returns a `number` of value `3`:

```
get-years-from-date-difference('2003-10-30','1999-11-28')
```

**format-date**

Returns the date for a given date and locale.

*Syntax*

```
format-date(strDate, "language", "country", "variant", "timezone")
```

*Parameters*

`strDate` is a `string` that represents the date in the format yyyy-mm-dd.

You can also specify zero or one set of parameters that identifies the locale of the equivalent date:

- `language` is a string that identifies the language of the locale.
- `country` is a string that identifies the country of the locale.
- `variant` identifies a vendor or web browser.
- `timezone` identifies the time zone of the locale.

  *For more information about date and time parameters, see Dateand time parameters.*

*Returns*

A `string` that represents the equivalent date in the format `yyyy-mm-dd`.

*Example*

The following expression returns the current date for the Pacific time zone in the United States, in a format that is appropriate for the Windows operating system:

```
format-date(currentdate(),"en", "US", "WIN", "US/Pacific-New")
```

### format-dateTime

Returns a string that represents the date and time in the format that is specific to a given locale and time zone.

*Syntax*

```
format-dateTime(strDateTime, (language, country, variant, timezone)?)
```

*Parameters*

`strDateTime` is a `string` that represents the date and time in the format `yyyy-mm-ddThh:mm:ssZ`.

You can also specify zero or one set of parameters that identifies the locale of the equivalent date:

- `language` is a string that identifies the language of the locale.
- `country` is a string that identifies the country of the locale.
- `variant` identifies a vendor or web browser.
- `timezone` identifies the time zone of the locale.

  *For more information about date and time parameters, see Dateand time parameters.*

*Returns*

A `string` that represents the localized date and time in the format `yyyy-mm-ddThh:mm:ssZ`.

### parse-date

Translates a given date to a given locale and time zone.

*Syntax*

```
parse-date(strDate, (language, country, variant, timezone)?)
```

*Parameters*

`strDate` is a `string` that represents the date in the format `yyyy-mm-dd`.

You can also specify zero or one set of parameters that identifies the locale of the equivalent date:

- `language` is a string that identifies the language of the locale.
- `country` is a string that identifies the country of the locale.
- `variant` identifies a vendor or web browser.
- `timezone` identifies the time zone of the locale.

  *For more information about date and time parameters, see Dateand time parameters.*

*Returns*

A `string` that represents the equivalent date in the format `yyyy-mm-dd`.

### parse-dateTime

Translates a given date and time to a given locale and time zone.

*Syntax*

```
parse-dateTime(strDateTime, (language, country, variant, timezone)?)
```

*Parameters*

`strDateTime` is a `string` that represents the date and time in the format `yyyy-mm-ddThh:mm:ssZ`.

You can also specify zero or one set of parameters that identifies the locale of the equivalent date:

- `language` is a string that identifies the language of the locale.
- `country` is a string that identifies the country of the locale.
- `variant` identifies a vendor or web browser.
- `timezone` identifies the time zone of the locale.

  *For more information about date and time parameters, see Dateand time parameters.*

*Returns*

A `string` that represents the translated date and time in the format `yyyy-mm-ddThh:mm:ssZ`.

### time-equal

Compares two given time values to see if they are equal.

*Syntax*

```
time-equal(strTime1, strTime2)
```

*Parameters*

`strTime1` is a `string` that represents one time to compare and is specified in the format `hh:mm:ss`.

`strTime2` is a `string` that represents the other time to compare and is specified in the format `hh:mm:ss`.

*Returns*

A Boolean value of `true` if the given time values are equal, otherwise `false`.

### time-greater-than

Compares two given time values to see if one is greater than the other.

*Syntax*

```
time-greater-than(strTime1, strTime2)
```

*Parameters*

`strTime1` is a `string` that represents one time to compare and is specified in the format `hh:mm:ss`.

`strTime2` is a `string` that represents the other time to compare and is specified in the format `hh:mm:ss`.

*Returns*

A Boolean value of `true` if the time that `strTime1` represents is greater than the time that `strDate2` represents, otherwise `false`.

**time-less-than**

Compares two given time values to see if one is less than the other.

*Syntax*

```
time-less-than(strTime1, strTime2)
```

*Parameters*

`strTime1` is a `string` that represents one time to compare and is specified in the format `hh:mm:ss`.

`strTime2` is a `string` that represents the other time to compare and is specified in the format `hh:mm:ss`.

*Returns*

A Boolean value of `true` if the time that `strTime1` represents is less than the time that `strDate2` represents, otherwise `false`.

## Document functions

This section describes functions that operate on `document` values that are available in expressions.

**getDocLength**

Retrieves the file size of a document.

*Syntax*

```
getDocLength(com.adobe.idp.Document)
```

*Parameters*

`com.adobe.idp.Document` is a reference to a serialized document.

*Returns*

A `Long` that contains the file size of the document in bytes. Returns `0` if the parameter does not evaluate to a document.

*Example*

A variable named `vardocument` references a PDF file that was serialized after being submitted as a work item form. The following example returns `455343`, the size of the referenced document in bytes:

```
getDocLength(/process_data/@vardocument)
```

**getDocAttribute**

Retrieves the value of the specified document attribute.

*Syntax*

```
getDocAttribute(com.adobe.idp.Document, String)
```

*Parameters*

`com.adobe.idp.Document` is a reference to a serialized document. `String` is a string that holds the name of the document attribute to retrieve. For information about available attributes, see Documentattributes for attachments and notes.

*Returns*

The value of the attribute specified by `String`. The attribute determines the type of value returned. Returns an empty string if the `com.adobe.idp.Document` parameter does not evaluate to a `com.adobe.idp.Document` value. Returns `null` if `String` does not hold a valid attribute name.

*Example*

A `document` variable named `varattachment` references a PDF file that was attached to a task. The following example returns an `integer` that represents the access permissions of the attachment:

```
getDocAttribute(/process_data/@varattachment,"wspermission")
```

**setDocAttribute**

Sets the value of a document attribute.

*Syntax*

```
setDocAttribute(com.adobe.idp.Document, String, String)
```

*Parameters*

`com.adobe.idp.Document` is a reference to a serialized document. The first `String` parameter is a string that holds the name of the document attribute to set. The second `String` parameter is a string that holds the value to set the document attribute to. For information about available attributes, see Document attributesfor attachments and notes.

*Returns*

A reference to the input `com.adobe.idp.Document` value with the attribute specified by the first `String` parameter set to the value specified by the second `String` parameter.

Returns a reference to an empty com.adobe.idp.Document value with the specified attribute set to the specified value if the first parameter does not evaluate to a document.

Returns a reference to the unchanged com.adobe.idp.Document value if the second or third parameters do not evaluate to a `String`.

*Example*

The following example returns the description of a document that the `document` variable named `vardoc1` references:

```
setDocAttribute(/process_data/@vardoc1,"wsdescription",
"An example description.")
```

**getDocContentType**

Retrieves the content type of a document. The content type identifies the type of the document, similar to the `mime-type` property of documents that are sent over the web. For example, a PDF document has a document type of `application/pdf`.

The document type of a document can have no value or can have any value that is set with the `setDocContent` function, when the document was attached to a task. For more information, see setDocContentType.

*Syntax*

```
getDocContentType(com.adobe.idp.Document)
```

*Parameters*

`com.adobe.idp.Document` is a serialized document that is referenced by a `document` variable.

*Returns*

A `String` that contains the content type. Returns an empty `String` if the parameter does not evaluate to a document.

**setDocContentType**

Sets the content type of a document. The content type identifies the type of the document, similar to the `mime-type` property of documents that are sent over the web. For example, a PDF document has a document type of `application/pdf`.

*Syntax*

```
setDocContentType(com.adobe.idp.Document, String)
```

*Parameters*

`com.adobe.idp.Document` is a serialized document that is referenced by a `document` variable.

`String` is a string that contains the content type that you want to set for the document.

*Returns*

A reference to a copy of the `com.adobe.idp.Document` value that was used as a parameter. Returns a new, empty document with the specified content type if the first parameter does not evaluate to a `com.adobe.idp.Document`.

**getDocContentBase64**

Retrieves the content of a document as a Base64-encoded string.

*Syntax*

```
getDocContentBase64(aDocument)
```

*Parameters*

`aDocument` is a serialized document value. You can reference the document value using an XPath expression.

*Returns*

A string value that represents the document. If the parameter does not evaluate to a document value, an empty string is returned.

*Example*

A text file is stored in a document variable named `documentVar`. The content of the text file is the single word *content*. The following expression returns the string value `Y29udGVudA==`:

```
getDocConententBase64(/process_data/@documentVar)
```

**getDocFromBase64**

Converts a Base64-encoded string to a document.

*Syntax*

```
getDocFromBase64(aString)
```

*Parameters*

`aString` is a string value that contains Base64-encoded text. You can reference the string value using an XPath expression.

*Returns*

A document value that represents a document, and the content of the document is the decoded text that `aString` contained.

*Example*

A string variable named `stringVar` contains the Base64-encoded text `Y29udGVudA==`. The following expression returns a document that contains the text *content*:

```
getDocFromBase64(/process_data/@stringVar)
```

## Miscellaneous

This section describes miscellaneous functions that are available in expressions.

### deserialize

Converts the text of a given string into an XML document.

*Syntax*

```
deserialize(string)
```

*Parameters*

`string` is an XPath expression that evaluates to a string value.

*Returns*

An `org.w3c.dom.Document` value that represents the XML document.

### is-null

Determines whether a node in the process data model exists.

**TIP:** In XPath, null values are represented as empty string values.

*Syntax*

```
is-null(node)
```

*Parameters*

`node` is an XPath expression that evaluates to a node in the process data model.

*Returns*

A `boolean` value of `true` if the node does not exist, and `false` if the node exists.

*Example*

The name of a string variable that a process includes is stringVar. The following expression returns `false`:

```
is-null(/process_data/@stringVar)
```

### serialize

Returns the string representation of given XML code. You can specify whether the string includes the XML declaration as the first line.

*Syntax*

```
serialize(node,bOmitXML)
```

*Parameters*

A `node` and an optional boolean`bOmitXML`. `node` is an XPath expression that evaluates to a data location that stores XML data. If a value of `true()` is provided for `bOmitXML`, the first line in the string representation includes the XML declaration. If `bOmitXML` is `false()`, the XML declaration is not included. If `bOmitXML` is not specified, the XML declaration is provided.

*Returns*

A `string` that represents the contents of the given node.

## Node set functions

This section describes the functions that operate on schema nodes that are available in expressions.

### count

Returns the number of nodes in a given node set.

*Syntax*

```
count(node-set)
```

*Parameters*

`node-set` is the path to a node in the process data model.

*Returns*

A `number` that holds the number of nodes.

### last

Returns a number equal to the context size from the expression evaluation context.

*Syntax*

```
last()
```

*Parameters*

None.

*Returns*

`number`.

**position**

Returns a number equal to the context position from the expression evaluation context.

*Syntax*

```
position()
```

*Parameters*

None.

*Returns*

`number`.

**sum**

Returns the sum, for each node in a given node-set, of the result of converting the string-values of the node to a number.

*Syntax*

```
sum(node-set)
```

*Parameters*

`node-set` is a node in the XML.

*Returns*

`number`.

## Number functions

This section describes the number functions that are available in expressions.

**ceiling**

Returns the smallest integer value that is not less than a given numeric value. Values are smaller in the direction of negative infinity.

The ceiling function returns the smallest (closest to negative infinity) number that is not less than the argument and that is an integer.

*Syntax*

```
ceiling(Expression)
```

*Parameters*

`Expression` is an expression that returns a numeric value.

*Returns*

An `integer` value.

**floor**

Returns the largest integer value that is not greater than a given numeric value. Values are larger in the direction of positive infinity.

*Syntax*

```
floor(number)
```

*Parameters*

`number` is a numeric value.

*Returns*

An `integer` value.

**number**

The number function converts a given numeric value to a number.

*Syntax*

```
number(object?)
```

*Parameters*

Zero or one objects.

*Returns*

A `number` value.

**round**

Returns a whole number that is closest to a given numeric value.

*Syntax*

```
round(number)
```

*Parameters*

`number` is any numeric value:

- If `number` is NaN, NaN is returned.
- If `number` is positive infinity, positive infinity is returned.
- If `number` is negative infinity, negative infinity is returned.
- If `number` is positive zero, positive zero is returned.
- If `number` is negative zero, negative zero is returned.
- If `number` is less than zero and greater than or equal to -0.5, negative zero is returned.
- If `number` is an equal distance from two different numbers, the number that is closest to positive infinity is returned.

*Returns*

An `integer` value.

## String functions

This section describes the string functions that are available in expressions.

### concat

Concatenates two or more given strings.

*Syntax*

```
concat(string, string, string*)
```

*Parameters*

Two or more string values, separated by commas.

*Returns*

String.

### contains

Checks whether a given string contains a second given string.

*Syntax*

```
contains(string, string)
```

*Parameters*

Two string values separated by commas. `contains` checks if the first string contains the second string.

*Returns*

A Boolean value of `True` if the first parameter contains the second parameter, otherwise `False`.

### ends-with

Checks that the last characters of a given string match the characters of another given string.

*Syntax*

```
ends-with(string, string)
```

*Parameters*

Two string values separated by a comma. `ends-with` checks that the first parameter starts with the second parameter.

*Returns*

A `boolean` value of `true` if the first parameter starts with the second parameter, otherwise `false`.

### lower-case

Converts all uppercase characters in a given string to the lowercase equivalent, if one exists.

*Syntax*

```
lower-case(string)
```

*Parameters*

A string value.

*Returns*

String.

### normalize-space

Removes leading and trailing white space characters from the given string and also replaces consecutive white space characters with a single space character.

*Syntax*

```
normalize-space(string?)
```

*Parameters*

Zero or one string to normalize.

*Returns*

String.

### starts-with

Checks that the first characters of a given string match the characters of another given string.

*Syntax*

```
starts-with(string, string)
```

*Parameters*

Two string values separated by a comma. `starts-with` checks that the first parameter starts with the second parameter.

*Returns*

A Boolean value of `True` if the first parameter starts with the second parameter, otherwise `False`.

**string**

Converts an object to a string.

*Syntax*

```
string(object?)
```

*Parameters*

Zero or one objects.

*Returns*

String.

**substring-after**

Checks that a given string contains the characters of a second given string and returns the part of the first string that follows the occurrence of the second string.

*Syntax*

```
substring-after(string, string)
```

*Parameters*

Two string values separated by a comma. `substring-after` checks that the first string contains the second string.

*Returns*

A string that contains the characters from the first parameter that follow the occurrence of the second parameter in the first parameter. If the first parameter does not include the characters from the second parameter, `substring-after` returns an empty string.

*Example*

The following expression returns `04/01`:

```
substring-after("2005/04/01","/")
```

### substring-before

Checks that a given string contains the characters of a second given string and returns the part of the first string that precedes the occurrence of the second string.

*Syntax*

```
substring-before(string, string)
```

*Parameters*

Two string values separated by a comma. `substring-before` checks that the first string contains the second string.

*Returns*

A string that contains the characters from the first parameter that precede the occurrence of the second parameter in the first parameter. If the first parameter does not include the characters from the second parameter, `substring-before` returns an empty string.

*Example*

The following expression returns `2005`:

```
substring-before("2005/04/01","/")
```

### substring

Returns a substring of a given string. The substring is specified by the index of the given string where the substring begins and the length of the substring. The length can be specified either explicitly or as the remaining characters after the beginning of the substring.

*Syntax*

```
substring(string, number, number?)
```

*Parameters*

A string and one or more numbers separated by commas. The string is the parameter from which `the`substring is extracted. The first number is the index of the string parameter where the substring begins. The second number, if provided, is the length of the substring. If the third parameter is not provided, the substring includes the characters from the index to the end of the given string.

*Returns*

String.

*Example*

For example, the first of the following two expressions returns `234`, and the second expression returns `6789`:

```
substring("123456789",2,3)
substring("123456789",6)
```

**string-length**

Calculates the number of characters in the given string.

*Syntax*

```
string-length(string?)
```

*Parameters*

Zero or one strings.

*Returns*

Number.

**translate**

In a given string, replaces the occurrence of any of the characters of a second given string with the corresponding characters of a third given string or removes the occurrence of characters of a second given string if no corresponding character exists in the third given string.

*Syntax*

```
translate(string, string, string)
```

*Parameters*

Three `string` values separated by commas. The first string is the string in which characters are replaced. The second string contains the characters whose occurrences in the first string are replaced. The third string contains the replacement characters that correspond with the characters in the second string.

The characters in the second string correspond with the characters in the third string that have the same index:

- If the second string is longer than the third string, the characters in the second string that have an index that is greater than the length of the third string have no corresponding characters.
- If the third string is longer than the second string, the characters in the third string that have an index that is greater than the length of the second string are ignored.
- If a character occurs more than once in the second string, the first occurrence of the character determines how that character is replaced in the first string.

*Returns*

A string that contains the first parameter with the replaced characters.

*Example*

The following expression includes a second parameter that includes characters that all correspond with a character in the third parameter. The expression returns `BAr`:

```
translate("bar","abc","ABC")
```

The following expression includes a second parameter that is longer than the third parameter. The expression returns `AAA`:

```
translate("--aaa--","abc-","ABC")
```

The following expression includes a second parameter that includes multiple occurrences of a character. The expression returns `BAr`:

```
translate("bar", "aba", "ABC")
```

**upper-case**

Converts all lowercase characters in a given string to the uppercase equivalent, if one exists.

*Syntax*

```
upper-case(string)
```

*Parameters*

One string.

*Returns*

```
string
```

## Operators

The following table provides the meaning of each operator that is available in expressions.

| Operator | Meaning |
|---|---|
| `and` | logical AND |
| `or` | logical OR |
| `>` | greater than |
| `>=` | greater than or equal to |
| `<` | less than |
| `<=` | less than or equal to |

| Operator | Meaning |
|---|---|
| `!=` | does not equal |
| `=` | equals |
| `div` | division |
| `mod` | modulus |
| `*` | multiply |
| `+` | add |
| `-` | subtract |

## Date and time parameters

The following table describes the values you can specify for parameters that are related to expressing a date, a time, or information about a locale.

| Parameter | Description |
|---|---|
| `strDate` | A string that represents a calendar date in the format `yyyy-mm-dd`:<br>• *`yyyy`* is the year.<br>*`mm` is the number that represents the month.*<br>*`dd` is the calendar day of the month.* |
| `strTime` | A string that represents the time in the format `hh:mm:ssZ`:<br>• *`hh`* is the hour, using the 24-hour format.<br>*`mm` is the minutes*<br>*`ss` is the seconds*<br>*`Z` is the time zone. See the `time zone` parameter in this table for more information.* |
| `strDateTime` | A string that represents the date and time in the format yyyy-mm-ddThh:mm:ss. |
| `country` | A two-letter code that identifies a country, as specified by ISO 3166. For example, FR identifies France, and US identifies the United States of America. |
| `language` | A two-letter code that identifies a language, as specified by ISO 639. For example `en` identifies the English language and `fr` identifies the French language. |

| Parameter | Description |
|---|---|
| `time zone` | A sequence of characters that identifies the time zone. You can express the time zone either as a value relative to the Greenwich Mean Time (GMT) or using a descriptive identifier.<br>Time zones relative to GMT are expressed in the format `GMTsignhh:mm`:<br>• *`sign` is either + or − and indicates whether the time zone is ahead of GMT (+) or behind GMT (−).*<br>*`hh` is the difference, in hours, between the time zone and GMT.*<br>*`mm` is the number of minutes, in addition to the hours, that the time zone differs from GMT.*<br>For example, a time zone that is five hours ahead of GMT is expressed as `GMT+05:00`.<br>To see a list of descriptive identifiers that you can use instead to express the time zone, see Timezone descriptive identifiers. |
| `variant` | A code that specifies a vendor or web browser. For example, `WIN` specifies Microsoft® Windows® and `MAC` specifies Apple® Macintosh®. To specify two variants, separate them with an underscore and put the most important variant first. |

## Time zone descriptive identifiers

The following descriptive identifiers are supported for representing the time zone in XPath expressions.

**A**

ACT

AET

Africa/Abidjan

Africa/Accra

Africa/Addis_Ababa

Africa/Algiers

Africa/Asmera

Africa/Bamako

Africa/Bangui

Africa/Banjul

Africa/Bissau

Africa/Blantyre

Africa/Brazzaville

Africa/Bujumbura

Africa/Cairo

Africa/Casablanca

Africa/Ceuta

Africa/Conakry

Africa/Dakar

Africa/Dar_es_Salaam

Africa/Djibouti

Africa/Douala

Africa/El_Aaiun

Africa/Freetown

Africa/Gaborone

Africa/Harare

Africa/Johannesburg

Africa/Kampala

Africa/Khartoum

Africa/Kigali

Africa/Kinshasa

Africa/Lagos

Africa/Libreville

Africa/Lome

Africa/Luanda

Africa/Lubumbashi

Africa/Lusaka

Africa/Malabo

Africa/Maputo

Africa/Maseru

Africa/Mbabane

Africa/Mogadishu

Africa/Monrovia

Africa/Nairobi

Africa/Ndjamena

Africa/Niamey

Africa/Nouakchott

Africa/Ouagadougou

Africa/Porto-Novo

Africa/Sao_Tome

Africa/Timbuktu

Africa/Tripoli

Africa/Tunis

Africa/Windhoek

AGT

America/Adak

America/Anchorage

America/Anguilla

America/Antigua

America/Araguaina

America/Argentina/Buenos_Aires

America/Argentina/Catamarca

America/Argentina/ComodRivadavia

America/Argentina/Cordoba

America/Argentina/Jujuy

America/Argentina/La_Rioja

America/Argentina/Mendoza

America/Argentina/Rio_Gallegos

America/Argentina/San_Juan

America/Argentina/Tucuman

America/Argentina/Ushuaia

America/Aruba

America/Asuncion

America/Atka

America/Bahia

America/Barbados

America/Belem

America/Belize

America/Boa_Vista

America/Bogota

America/Boise

America/Buenos_Aires

America/Cambridge_Bay

America/Campo_Grande

America/Cancun

America/Caracas

America/Catamarca

America/Cayenne

America/Cayman

America/Chicago

America/Chihuahua

America/Coral_Harbour

America/Cordoba

America/Costa_Rica

America/Cuiaba

America/Curacao

America/Danmarkshavn

America/Dawson

America/Dawson_Creek

America/Denver

America/Detroit

America/Dominica

America/Edmonton

America/Eirunepe

America/El_Salvador

America/Ensenada

America/Fort_Wayne

America/Fortaleza

America/Glace_Bay

America/Godthab

America/Goose_Bay

America/Grand_Turk

America/Grenada

America/Guadeloupe

America/Guatemala

America/Guayaquil

America/Guyana

America/Halifax

America/Havana

America/Hermosillo

America/Indiana/Indianapolis

America/Indiana/Knox

America/Indiana/Marengo

America/Indiana/Vevay

America/Indianapolis

America/Inuvik

America/Iqaluit

America/Jamaica

America/Jujuy

America/Juneau

America/Kentucky/Louisville

America/Kentucky/Monticello

America/Knox_IN

America/La_Paz

America/Lima

America/Los_Angeles

America/Louisville

America/Maceio

America/Managua

America/Manaus

America/Martinique

America/Mazatlan

America/Mendoza

America/Menominee

America/Merida

America/Mexico_City

America/Miquelon

America/Monterrey

America/Montevideo

America/Montreal

America/Montserrat

America/Nassau

America/New_York

America/Nipigon

America/Nome

America/Noronha

America/North_Dakota/Center

America/Panama

America/Pangnirtung

America/Paramaribo

America/Phoenix

America/Port_of_Spain

America/Port-au-Prince

America/Porto_Acre

America/Porto_Velho

America/Puerto_Rico

America/Rainy_River

America/Rankin_Inlet

America/Recife

America/Regina

America/Rio_Branco

America/Rosario

America/Santiago

America/Santo_Domingo

America/Sao_Paulo

America/Scoresbysund

America/Shiprock

America/St_Johns

America/St_Kitts

America/St_Lucia

America/St_Thomas

America/St_Vincent

America/Swift_Current

America/Tegucigalpa

America/Thule

America/Thunder_Bay

America/Tijuana

America/Toronto

America/Tortola

America/Vancouver

America/Virgin

America/Whitehorse

America/Winnipeg

America/Yakutat

America/Yellowknife

Antarctica/Casey

Antarctica/Davis

Antarctica/DumontDUrville

Antarctica/Mawson

Antarctica/McMurdo

Antarctica/Palmer

Antarctica/Rothera

Antarctica/South_Pole

Antarctica/Syowa

Antarctica/Vostok

Arctic/Longyearbyen

ART

Asia/Aden

Asia/Almaty

Asia/Amman

Asia/Anadyr

Asia/Aqtau

Asia/Aqtobe

Asia/Ashgabat

Asia/Ashkhabad

Asia/Baghdad

Asia/Bahrain

Asia/Baku

Asia/Bangkok

Asia/Beirut

Asia/Bishkek

Asia/Brunei

Asia/Calcutta

Asia/Choibalsan

Asia/Chongqing

Asia/Chungking

Asia/Colombo

Asia/Dacca

Asia/Damascus

Asia/Dhaka

Asia/Dili

Asia/Dubai

Asia/Dushanbe

Asia/Gaza

Asia/Harbin

Asia/Hong_Kong

Asia/Hovd

Asia/Irkutsk

Asia/Istanbul

Asia/Jakarta

Asia/Jayapura

Asia/Jerusalem

Asia/Kabul

Asia/Kamchatka

Asia/Karachi

Asia/Kashgar

Asia/Katmandu

Asia/Krasnoyarsk

Asia/Kuala_Lumpur

Asia/Kuching

Asia/Kuwait

Asia/Macao

Asia/Macau

Asia/Magadan

Asia/Makassar

Asia/Manila

Asia/Muscat

Asia/Nicosia

Asia/Novosibirsk

Asia/Omsk

Asia/Oral

Asia/Phnom_Penh

Asia/Pontianak

Asia/Pyongyang

Asia/Qatar

Asia/Qyzylorda

Asia/Rangoon

Asia/Riyadh

Asia/Riyadh87

Asia/Riyadh88

Asia/Riyadh89

Asia/Saigon

Asia/Sakhalin

Asia/Samarkand

Asia/Seoul

Asia/Shanghai

Asia/Singapore

Asia/Taipei

Asia/Tashkent

Asia/Tbilisi

Asia/Tehran

Asia/Tel_Aviv

Asia/Thimbu

Asia/Thimphu

Asia/Tokyo

Asia/Ujung_Pandang

Asia/Ulaanbaatar

Asia/Ulan_Bator

Asia/Urumqi

Asia/Vientiane

Asia/Vladivostok

Asia/Yakutsk

Asia/Yekaterinburg

Asia/Yerevan

AST

Atlantic/Azores

Atlantic/Bermuda

Atlantic/Canary

Atlantic/Cape_Verde

Atlantic/Faeroe

Atlantic/Jan_Mayen

Atlantic/Madeira

Atlantic/Reykjavik

Atlantic/South_Georgia

Atlantic/St_Helena

Atlantic/Stanley

Australia/ACT

Australia/Adelaide

Australia/Brisbane

Australia/Broken_Hill

Australia/Canberra

Australia/Currie

Australia/Darwin

Australia/Hobart

Australia/LHI

Australia/Lindeman

Australia/Lord_Howe

Australia/Melbourne

Australia/North

Australia/NSW

Australia/Perth

Australia/Queensland

Australia/South

Australia/Sydney

Australia/Tasmania

Australia/Victoria

Australia/West

Australia/Yancowinna

**B**

BET

Brazil/Acre

Brazil/DeNoronha

Brazil/East

Brazil/West

BST

**C**

Canada/Atlantic

Canada/Central

Canada/Eastern

Canada/East-Saskatchewan

Canada/Mountain

Canada/Newfoundland

Canada/Pacific

Canada/Saskatchewan

Canada/Yukon

CAT

CET

Chile/Continental

Chile/EasterIsland

CNT

CST

CST6CDT

CTT

Cuba

**E**

EAT

ECT

EET

Egypt

Eire

EST

EST5EDT

Etc/GMT

Etc/GMT+0

Etc/GMT+1

Etc/GMT+10

Etc/GMT+11

Etc/GMT+12

Etc/GMT+2

Etc/GMT+3

Etc/GMT+4

Etc/GMT+5

Etc/GMT+6

Etc/GMT+7

Etc/GMT+8

Etc/GMT+9

Etc/GMT0

Etc/GMT-0

Etc/GMT-1

Etc/GMT-10

Etc/GMT-11

Etc/GMT-12

Etc/GMT-13

Etc/GMT-14

Etc/GMT-2

Etc/GMT-3

Etc/GMT-4

Etc/GMT-5

Etc/GMT-6

Etc/GMT-7

Etc/GMT-8

Etc/GMT-9

Etc/Greenwich

Etc/UCT

Etc/Universal

Etc/UTC

Etc/Zulu

Europe/Amsterdam

Europe/Andorra

Europe/Athens

Europe/Belfast

Europe/Belgrade

Europe/Berlin

Europe/Bratislava

Europe/Brussels

Europe/Bucharest

Europe/Budapest

Europe/Chisinau

Europe/Copenhagen

Europe/Dublin

Europe/Gibraltar

Europe/Helsinki

Europe/Istanbul

Europe/Kaliningrad

Europe/Kiev

Europe/Lisbon

Europe/Ljubljana

Europe/London

Europe/Luxembourg

Europe/Madrid

Europe/Malta

Europe/Mariehamn

Europe/Minsk

Europe/Monaco

Europe/Moscow

Europe/Nicosia

Europe/Oslo

Europe/Paris

Europe/Prague

Europe/Riga

Europe/Rome

Europe/Samara

Europe/San_Marino

Europe/Sarajevo

Europe/Simferopol

Europe/Skopje

Europe/Sofia

Europe/Stockholm

Europe/Tallinn

Europe/Tirane

Europe/Tiraspol

Europe/Uzhgorod

Europe/Vaduz

Europe/Vatican

Europe/Vienna

Europe/Vilnius

Europe/Warsaw

Europe/Zagreb

Europe/Zaporozhye

Europe/Zurich

**G**

GB

GB-Eire

GMT

GMT0

Greenwich

**H**

Hongkong

HST

**I**

Iceland

IET

Indian/Antananarivo

Indian/Chagos

Indian/Christmas

Indian/Cocos

Indian/Comoro

Indian/Kerguelen

Indian/Mahe

Indian/Maldives

Indian/Mauritius

Indian/Mayotte

Indian/Reunion

Iran

Israel

IST

**J**

Jamaica

Japan

JST

**K**

Kwajalein

**L**

Libya

**M**

MET

Mexico/BajaNorte

Mexico/BajaSur

Mexico/General

Mideast/Riyadh87

Mideast/Riyadh88

Mideast/Riyadh89

MIT

MST

MST7MDT

**N**

Navajo

NET

NST

NZ

NZ-CHAT

**P**

Pacific/Apia

Pacific/Auckland

Pacific/Chatham

Pacific/Easter

Pacific/Efate

Pacific/Enderbury

Pacific/Fakaofo

Pacific/Fiji

Pacific/Funafuti

Pacific/Galapagos

Pacific/Gambier

Pacific/Guadalcanal

Pacific/Guam

Pacific/Honolulu

Pacific/Johnston

Pacific/Kiritimati

Pacific/Kosrae

Pacific/Kwajalein

Pacific/Majuro

Pacific/Marquesas

Pacific/Midway

Pacific/Nauru

Pacific/Niue

Pacific/Norfolk

Pacific/Noumea

Pacific/Pago_Pago

Pacific/Palau

Pacific/Pitcairn

Pacific/Ponape

Pacific/Port_Moresby

Pacific/Rarotonga

Pacific/Saipan

Pacific/Samoa

Pacific/Tahiti

Pacific/Tarawa

Pacific/Tongatapu

Pacific/Truk

Pacific/Wake

Pacific/Wallis

Pacific/Yap

PLT

PNT

Poland

Portugal

PRC

PRT

PST

PST8PDT

**R**

ROK

**S**

Singapore

SST

SystemV/AST4

SystemV/AST4ADT

SystemV/CST6

SystemV/CST6CDT

SystemV/EST5

SystemV/EST5EDT

SystemV/HST10

SystemV/MST7

SystemV/MST7MDT

SystemV/PST8

SystemV/PST8PDT

SystemV/YST9

SystemV/YST9YDT

**T**

Turkey

**U**

UCT

Universal

US/Alaska

US/Aleutian

US/Arizona

US/Central

US/Eastern

US/East-Indiana

US/Hawaii

US/Indiana-Starke

US/Michigan

US/Mountain

US/Pacific

US/Pacific-New

US/Samoa

UTC

**V**

VST

**W**

WET

W-SU

**Z**

Zulu

# 16. Testing and troubleshooting process versions

Before you move your process version to a staging environment for rigorous testing, you should test it in the development environment. If you encounter errors during development testing, you need to investigate and debug the errors.

Various tools and strategies can be used for testing:

- Discover and correct design flaws as you create the process version.
- Observe the behavior of the process version at run time using sample input data.
- Invoke the process version by using scenarios that mimic realistic use cases.

You can also implement features in process versions that help to troubleshoot run-time errors.

RELATED LINKS:

Process execution

Handling errors in the production environment

Process designs for reuse

Process instances

## 16.1. Validation reports

Workbench automatically checks for problems and potential problems in the process design and displays the results as messages in the Validation Report view. To find and fix problems as they occur, look at the Validation Report view during the development of processes.

You typically check for validation messages after you perform the following tasks:

- Substantially change a process or other application asset. You check the validation messages to find any problems that you have introduced.
- Address existing problems. You check the validation messages to see if the problems are fixed.

The Validation Report provides tools for investigating validation messages.

You can also customize some of the behavior of the validation features.

For a description of the validation messages that can appear in the Validation Report view, see Validationmessage reference.

**NOTE:** When validating operation property values, problems can be detected only for properties that use simple data types as values. Simple data types are those that Foundation defines.

**NOTE:** Validation does not occur for property configurations of operations that use a property editor that the operation's service defines.

## Reviewing validation messages

Use the Validation Report view to read validation messages and investigate their causes. The Validation Report view provides three columns of information for each validation message:

**Description:**

Describes the validation issue and provides the severity level.

**Error Code:**

Provides the code that identifies the message.

**Asset:**

Identifies the asset to which the message applies.

**Path:**

Identifies the location of the asset to which the message applies.

**Location:**

Identifies the element in the process diagram or other asset that caused the validation message.

The severity level is either Error, Warning, or Information:

- Error messages indicate problems that must be fixed to enable the successful execution of the process version.

  **NOTE:** Workbench does not prevent you from deploying or executing assets that generate validation errors.
- Warning messages indicate potential problems that do not prevent the execution of the process but could cause unwanted behavior.
- Information messages indicate benign behavior.

The View Menu allows you to select how messages are grouped and which messages appear. Expand and collapse message groups using the plus sign beside the name of the group.

For a description of the validation messages that can appear in the Validation Report view, see Validationmessage reference.

The following procedures describe how to use the tools that the Validation Report view provides for investigating the cause of messages.

**To see documentation for a validation message:**

1) In the Validation Report view, select the message and then click the Help button or press F1.
2) In the Help view, click the link for the validation message.

**To change the way that validation messages are grouped:**

1) In the Validation Report view, click the View Menu button and select one of the following:
   – Group By > Severity
   – Group By > Application

**To change which validation messages appear:**

1) In the Validation Report view, click the View Menu button and select one of the following:
   – Show > All Errors (no warnings are displayed)
   – Show > Problems on Selection (only messages for the currently selected item are displayed)
   – Show > Show All

   **NOTE:** To hide warnings for assets that are not currently selected, select both All Errors and Problems on Selection.

**To display the location of the problem:**

1) In the Validation Report view, double-click the message. The asset that is the source of the problem appears or is selected. (For example, the element where the problem occurs is selected.) In some cases, a dialog box you can use to fix the problem appears.

**To copy validation messages:**

1) In the Validation Report view, click the Copy button. The text for all validation messages can then be pasted elsewhere, such as an email message.

**To refresh the validation messages for a process:**

If the validation messages do not reflect the current contents of your process, you can manually update the Validation Report view.

1) In the Application view, right-click the process and click Validate.

## Customizing validation behavior

Preferences can be configured to control some of the behavior of the validation features. The following table describes the preferences that you can configure.

| Preference | Default value | Description |
|---|---|---|
| `-Dcom.adobe.process.validation.warn.numactions` | 50 | The number of operations that causes warning WBP-105 to occur. (See WBP-105.) |
| `-Dcom.adobe.process.validation.warn.numvars` | 20 | The number of variables that causes warning WBP-104 to occur. (See WBP-104.) |

To configure preferences, modify the workbench.ini file. Each line in the file contains a preference setting in the following format:

```
preference_name = value
```

For example, the following preference setting specifies that the warning WBP-104 is generated when 20 variables are created for a process version:

```
-Dcom.adobe.process.validation.warn.num_variables=20
```

The workbench.ini file is located in the *[install directory]*/AEM forms Workbench/Workbench directory, where *[install directory]* is the location where you installed Workbench.

**To customize validation behavior:**

1) Open the workbench.ini file in a text editor.
2) Modify the values of preferences as required.
3) Save workbench.ini and restart Workbench.

# 16.2. Validation message reference

This section provides information about the validation messages that can appear in the Validation Report view.

**TIP:** To quickly get help about validation messages, in the Validation Report view, select the message and press F1.

## DCI-007

```
Action profile "action profile name" is out of sync with the service type
service: input parameter "parameter name" has been removed
```

This validation message applies to action profiles.

| Explanation | Resolution |
|---|---|
| An input variable for the prepare data, render, or submit service used by the action profile is missing. | • In the Application view, right-click the asset and click Manage Action Profiles. If any action profiles in the list are out of sync, you are prompted to update them. Alternatively, in the Validation Reports view, double-click the validation message to open the Manage Action Profiles dialog box. (See Modifyingand creating action profiles.)<br>*If it does not exist, create the missing input variable. (See Creatingvariables.)*<br>*Add the missing input process variable to the action profile. (See Createa prepare data, render, or submit process.)* |

## DCI-008

```
Action profile "action profile name" is out of sync with the service name
service: output parameter "parameter name" has been removed
```

This validation message applies to action profiles.

| Explanation | Resolution |
|---|---|
| The output variable for the prepare data, render, or submit service used by the action profile is missing. | • In the Application view, right-click the asset and click Manage Action Profiles. If any action profiles in the list are out of sync, you are prompted to update them. Alternatively, in the Validation Reports view, double-click the validation message to open the Manage Action Profiles dialog box. (See Modifyingand creating action profiles.)<br>*If it does not exist, create the missing output variable. (See Creating variables.)*<br>*Add the missing output process variable to the action profile. (See Createa prepare data, render, or submit process.)* |

## DCI-009

```
Action profile "action profile name" is out of sync with the service name
service: "parameter name" is no longer an input parameter
```

This validation message applies to action profiles.

| Explanation | Resolution |
|---|---|
| An input variable for the prepare data, render, or submit service used by the action profile has been changed to an output parameter. | • In the Application view, right-click the asset and click Manage Action Profiles. If any action profiles in the list are out of sync, you are prompted to update them. Alternatively, in the Validation Reports view, double-click the validation message to open the Manage Action Profiles dialog box. (See Modifyingand creating action profiles.)<br>*Using the Variables view, open the properties for the variable and change the Purpose option to Input. (See Creating-variables.)* |

## DCI-010

`Action profile "`*`action profile name`*`" is out of sync with the `*`service name`*` service: "`*`parameter name`*`" is no longer an output parameter`

This validation message applies to action profiles.

| Explanation | Resolution |
|---|---|
| An output variable for the prepare data, render, or submit service used by the action profile has been changed to an input parameter. | • In the Application view, right-click the asset and click Manage Action Profiles. If any action profiles in the list are out of sync, you are prompted to update them. Alternatively, in the Validation Reports view, double-click the validation message to open the Manage Action Profiles dialog box. (See Modifyingand creating action profiles.)<br>*Using the Variables view, open the properties for the variable and change the Purpose option to Output. (See Creating-variables.)* |

## DCI-011

`Action profile "`*`action profile name`*`" is out of sync with the `*`service name`*` service: input parameter "`*`parameter`*` name" has been added`

This validation message applies to action profiles.

| Explanation | Resolution |
|---|---|
| An input variable has been added to the prepare data, render, or submit service used by the action profile. | • In the Application view, right-click the asset and click Manage Action Profiles. If any action profiles in the list are out of sync, you are prompted to update them. Alternatively, in the Validation Reports view, double-click the validation message to open the Manage Action Profiles dialog box. (See Modifyingand creating action profiles.) |

## DCI-012

```
Action profile "action profile name" is out of sync with the service name
service: output parameter "parameter name" has been added
```

This validation message applies to action profiles.

| Explanation | Resolution |
|---|---|
| An output variable has been added to the prepare data, render, or submit service used by the action profile. | • In the Application view, right-click the asset and click Manage Action Profiles. If any action profiles in the list are out of sync, you are prompted to update them. Alternatively, in the Validation Reports view, double-click the validation message to open the Manage Action Profiles dialog box. (See Modifyingand creating action profiles.) |

## DCI-013

```
service type service "service name" service of Action profile "action
profile name" does not exist
```

This validation message applies to action profiles.

| Explanation | Resolution |
|---|---|
| The prepare data, render, or submit service used by the action profile is missing. | • In the Application view, right-click the asset and click Manage Action Profiles. If any action profiles in the list are out of sync, you are prompted to update them. Alternatively, in the Validation Reports view, double-click the validation message to open the Manage Action Profiles dialog box. (See Modifyingand creating action profiles.)<br>*Modify the action profile to specify a process that exists, or select the default configuration for the service. (See Modifyingand creating action profiles.)* |

## WBP-001

`Process does not contain a start activity or event start point.`

This validation message applies to the process.

| Explanation | Resolution |
|---|---|
| The process has no entry point. | Perform one of the following tasks:<br>• Configure an operation as the process start point. (See Specifyingthe start activity of a process.)<br>*Add an event type as a start point for the process. (See Controllingprocess flow using events.)*<br>*Draw a route from an event start point to an operation. (See Drawingroutes to link operations.)* |

## WBP-003

`The event '`*`event name`*`' is not connected to any activity.`

This validation message applies to the process.

| Explanation | Resolution |
|---|---|
| No route begins or ends at the named event throw or receive. | Perform one of the following tasks:<br>• Delete the event throw or receive. (See Deletingevents.)<br>*Draw one or more routes (as required) to connect the event throw or receive to another operation. (See Drawingroutes to link operations.)* |

## WBP-007

`Event receive '`*`event_receive_name`*`' cannot exist in short lived process.`

This validation message applies to the process.

| Explanation | Resolution |
|---|---|
| Short-lived processes cannot include event receives because they execute synchronously. Event receives require asynchronous execution. (See Short-livedprocesses and long-lived processes.) | Perform one of the following tasks:<br>• Configure the process to be long-lived.<br>*If possible, remove the event receive. Redesign your process architecture so that a subprocess is invoked instead of executing an event throw and an event receive. (See Processdesigns for reuse.)* |

## WBP-008

`Input parameter '`*`property name`*`' for activity '`*`operation name`*`' is required and has no value specified.`

This validation message applies to an operation.

| Explanation | Resolution |
|---|---|
| No value is specified for the required property. | Use the Process Properties view to specify a value for the operation property. (See Inputand output data for operations.)<br>***TIP:*** *An asterisk (*) appears next to required properties in the Process Properties view.* |

## WBP-009

`The subprocess '`*`process name`*`' referenced by activity '`*`activity name`*`' does not exist.`

This validation message applies to an operation.

| Explanation | Resolution |
|---|---|
| The process includes the invoke operation of the service for the named subprocess, but the service for the subprocess does not exist.<br>The subprocess was either deactivated or deleted after the invoke operation was added. | Perform one of the following tasks:<br>• Delete the operation. (See Addingand deleting operations.)<br>*Deploy the application that contains the process that is being used as a subprocess.* |

## WBP-010

`The subprocess '`*`process name`*`' specifies an invocation policy 'Wait for response' that is not supported for a 'short-lived' process.`

This validation message applies to an operation.

| Explanation | Resolution |
|---|---|
| The process version is configured to be short-lived, and does not support waiting for the response of a subprocess.<br>At run time, a short-lived process executes synchronously. To continue executing after the subprocess completes constitutes asynchronous execution. | Perform one of the following tasks:<br>• Configure the process to be long-lived.<br>*Configure the invoke operation of the subprocess so that the Invocation Policy property is set to Do Not Wait For Response.*<br>***NOTE:*** *The operations that the process map includes can dictate the type of the process version or branch that you need to use. (See Processexecution.)* |

## WBP-013

`Activity '`*`operation name`*`' specifies more than one unconditional route.`

This validation message applies to a route.

| Explanation | Resolution |
|---|---|
| Multiple routes that begin at the named operation do not have conditions associated with them. All but one of these routes will never be followed during process execution. | Perform one of the following tasks:<br>• Create conditions for the routes. (See Makingdecisions using routes.)<br>*Alter the design of the process so that only one route begins at the operation.* |

## WBP-018

```
XPath is not syntactically correct for parameter 'property name' at
activity 'operation name'. [XPath exception message]
```

This validation message applies to an XPath expression.

| Explanation | Resolution |
|---|---|
| The value of the named property is an XPath expression that contains one or more syntax errors. The exception message describes why the error occurred. | Correct the expression to use correct XPath syntax. (See CreatingXPath expressions.) |

## WBP-019

```
XPath is not syntactically correct for expression 1 of condition
'condition number' on route 'route name'.
```

This validation message applies to an XPath expression.

| Explanation | Resolution |
|---|---|
| The named condition for the route includes a syntax error in the expression for the expr1 property. (See Routingcondition format.) | Specify an expression for the expr1 property of the condition. (See Addingand modifying routing conditions.) |

## WBP-020

```
XPath is not syntactically correct for expression 2 of condition
'condition number' on route 'route name'.
```

This validation message applies to an XPath expression.

| Explanation | Resolution |
|---|---|
| The named condition for the route includes a syntax error in the expression for the expr2 property. (See Routingcondition format.) | Specify an expression for the expr2 property of the condition. (See Addingand modifying routing conditions.) |

## WBP-033

```
Variable '%s' is of unknown datatype '%s'.
```

This is a variable validation message.

| Explanation | Resolution |
|---|---|
| The type of the named variable is not supported.<br>This error occurs when the component that provides support for the data type is uninstalled after the variable is created. | Perform one of the following tasks:<br>• Change the type of the variable. (See Editingvariables.)<br>*Install the component that provides support for the data type. (See Installingcomponents.)* |

## WBP-034

```
The event start point 'event start point name' references data from
variable 'variable name' that is not an input variable.
```

This validation message applies to an event.

| Explanation | Resolution |
|---|---|
| An XPath expression in the properties of the event start point is attempting to store event data in a process variable that is not an input variable. | Perform one of the following tasks:<br>• Change the expression so that it references an input variable. (See Creating XPathexpressions.)<br>*Modify the process variable so that it is an input variable. (See Editingvariables.)*<br>***TIP:*** *You can provide the required data in the event throw and use an event data map in the event start point to retrieve the data. (See Controllingprocess flow using events.)* |

## WBP-035

```
Input parameter 'property name' references non-existent variable
'variable name'.
```

This validation message applies to an operation.

| Explanation | Resolution |
|---|---|
| The value of the named input property is set to a variable that does not exist. The named variable was deleted after the property value was set. | Configure the property to use a different value. (See Inputand output data for operations.) |

## WBP-036

```
Unable to coerce from variable 'variable name' to input parameter
'property name' ('specified data type' to 'required data type').
```

This validation message applies to an operation.

| Explanation | Resolution |
|---|---|
| The data type of the named variable does not match the data type that the input property requires. The AEM forms Server may fail to convert the data to the required type at run time. | Configure the property to use a value of the correct type. (See Inputand output data for operations and Servicereference.) |

## WBP-037

```
Output parameter 'property name' references non-existent variable
'variable name'.
```

This validation message applies to an operation.

| Explanation | Resolution |
|---|---|
| The named output property of an operation is set to a variable that does not exist. The named variable was deleted after the property was configured. | Configure the property to use a different value. (See Inputand output data for operations.) |

## WBP-038

```
Unable to coerce from output parameter 'property name' to variable
'variable name' ('specified data type' to 'required data type').
```

This validation message applies to an operation.

| Explanation | Resolution |
|---|---|
| The data type of the named variable does not match the data type that the output property requires. The AEM forms Server may fail to convert the output data to the variable type at run time. | Configure the property to use a variable of the correct type. (See Input andoutput data for operations and Servicereference.) |

## WBP-039

```
Reference is made to undeclared variable 'variable name'.
```

This validation message applies to a variable.

| Explanation | Resolution |
|---|---|
| A reference is made to the named variable, but the variable does not exist.<br>The reference is made in an XPath expression. | Perform one of the following tasks:<br>• Create the variable that is referenced. (See Creatingvariables.)<br>*Modify the XPath expression so that it does not reference the undeclared variable. (See Buildingexpressions by using XPath Builder.)*<br>*Verify that the undeclared variable is correctly in the process. Variables are case sensitive. For example, referencing your variable as cmroot when you already defined it as cmRoot is an error.* |

## WBP-040

`Cannot assign a value to a read-only variable '`*`variable name`*`'.`

This validation message applies to a variable.

| Explanation | Resolution |
|---|---|
| An attempt is made to set the value of a configuration variable. The values of configuration variables cannot be changed at run time. | Perform one of the following tasks:<br>• Use a different variable to store the run time value. (See Creatingvariables.)<br>*Change the configuration variable to a process variable. (See Editing variables.)* |

## WBP-101

`Unconnected activity '`*`operation name`*`'.`

This validation message applies to the process.

| Explanation | Resolution |
|---|---|
| No route begins or ends at the named operation, or all routes that begin at the activity or operation end at the process start point. | Perform one of the following tasks:<br>• Draw one or more routes (as required) to connect the activity or operation to another activity or operation. (See Addin-gand deleting routes.)<br>*Ensure that a route that begins at the activ-ity or operation does not end at the start point of the process.* |

## WBP-102

`Abstract activity '`*`activity name`*`' has no runtime semantic.`

This validation message applies to the process.

| Explanation | Resolution |
|---|---|
| The named activity is not defined. | Specify a service operation to define the activity. (See Adding,defining, and deleting abstract activities.) |

## WBP-104

`Count of declared variables `*`total variables`*` exceeds recommendation.`

This validation message applies to the process.

| Explanation | Resolution |
|---|---|
| The number of variables that are created for the process version is greater than the recommended maximum number.<br>A large number of variables can negatively affect run-time performance of the AEM Forms Server. | Perform one or both of the following tasks, if possible:<br>• If possible, reduce the number of variables by reusing them.<br>*If possible, avoid using complex variable types.*<br>For more information, see Reuseof variables. |

The maximum recommended number of variables is specified in the Workbench preferences. (See Customizingvalidation behavior.)

## WBP-105

`Count of declared activities `*`total operations`*` exceeds recommendation.`

This validation message applies to the process.

| Explanation | Resolution |
|---|---|
| The number of activities or operations in the process diagram is greater than the recommended maximum number. | Perform one of the following tasks:<br>• Create multiple processes instead of a single process and link them at run time as subprocesses. (See Processdesigns for reuse.)<br>*Change the maximum number to suit your design architecture. (See Customizingvalidation behavior.)* |

The maximum recommended number of operations is specified in the Workbench preferences. (See Customizingvalidation behavior.)

## WBP-107

```
Action name 'operation name' is reused on multiple branches.  Name should
be unique within a Process.
```

This validation message applies to the process.

| Explanation | Resolution |
|---|---|
| Two or more operations that belong to different branches have the same value for the Name property.<br>Use names that are unique within the process to avoid confusion when identifying operations.<br>Operations that have the same name are difficult to distinguish when using the forms workflow tools in Administration Console, and in BAM Dashboard. | Change the Name property for the operations so that they are unique within the process version. (See Commonoperation properties.) |

## WBP-109

```
Dubious parameter coercion from variable 'variable name' to parameter
'property name' ('specified data type' to 'required data type').
```

This validation message applies to an operation.

| Explanation | Resolution |
|---|---|
| The data type of the named variable does not match the data type that the property requires. At run time, the AEM Forms Server will convert the variable data to the required type, but the results of the conversion may not be correct. | Perform one of the following tasks:<br>• Invoke the process and verify that the data conversion is correct. (See Recordingand playing back process *and* Monitoringvari-ables using the Variable Logger service.)<br>*Configure the property to use a variable of the correct type. (See Configuringinput and output data and Servicereference.)* |

## WBP-110

`Dubious variable coercion from output parameter '`*`property name`*`' to variable '`*`variable name`*`' ('`*`specified data`* `type' to '`*`required data type`*`').`

This validation message applies to an operation.

| Explanation | Resolution |
|---|---|
| The data type of the named variable does not match the data type that the output property requires. At run time, the AEM Forms Server will convert the output data to the variable type, but the results of the conversion may not be correct. | Perform one of the following tasks:<br>• Invoke the process and verify that the data conversion is correct. (See Recordingand playing back process *and* Monitoringvari-ables using the Variable Logger service.)<br>*Configure the property to use a variable of the correct type. (See Configuringinput and output data and Servicereference.)* |

## WBP-111

`Output of activity '`*`operation name`*`' is not assigned.`

This validation message applies to an operation.

| Explanation | Resolution |
|---|---|
| An output property of the named operation has not been configured. | If you want to save the output data, specify a data location, such as a variable, in which to store the Output property. (See Configuringinput and output data.) |

## WBP-112

`The subprocess '`*`process name`*`' referenced by activity '`*`operation name`*`' is not presently in the ACTIVE state.`

This validation message applies to an operation.

| Explanation | Resolution |
|---|---|
| The named operation invokes the named subprocess, and the subprocess is not activated. | Activate the application that contains the subprocess so that its service is available. |

## WBP-120

`XPath references namespace unspecified by the process's properties for parameter '`*`property name`*`' at activity '`*`operation name`*`'.`

This is an XPath validation message.

| Explanation | Resolution |
|---|---|
| The value of the named property is an XPath expression that contains an undefined namespace. | Perform one of the following tasks:<br>• Define the namespace in the process properties.<br>*Ensure that the namespace is spelled correctly in the XPath expression. (See CreatingXPath expressions.)*<br>*Remove the namespace from the XPath expression. (See Creating XPathexpressions.)* |

## WBP-150

`XPath expression: You might be using the true() XPath function to test for node existence. To test for node existence, compare true() with the results of using the boolean()function on the node.`

This is an XPath validation message.

| Explanation | Resolution |
|---|---|
| The node that an XPath expression evaluates to is being set or compared to the results of the `true` function. The `true` function returns a value of true.<br>To test for the existence of a node, use the `boolean` function. The following route condition tests whether the node for the `stringVar` variable exists:<br>`boolean(/process_data/@stringVar) = true()` | Ensure that you are using the `true` function correctly. (See true.)<br>To test the value of a boolean value, convert the value to a string. The following example converts the boolean variable named *boolVar* to a string, and compares it to 'true':<br>string(/process_data/@boolVar) = "true" |

## WBP-151

```
XPath expression: You might be using the false() XPath function to test
for node existence. To test for node existence, compare false() with the
results of using the boolean()function on the node.
```

This is an XPath validation message.

| Explanation | Resolution |
|---|---|
| The node that an XPath expression evaluates to is being set or compared to the results of the `false` function. The `false` function returns a value of false.<br>To test for the existence of a node, use the `boolean` function. The following route condition tests whether the node for the `stringVar` variable does not exist:<br>`boolean(/process_data/@stringVar) = false()` | Ensure that you are using the `false` function correctly. (See false.) |

## WBP-152

Service '*service name*' used by activity '*activity name*' is deprecated.

This validation message applies to a service.

| Explanation | Resolution |
|---|---|
| The activity is configured to use a service that AEM Forms does not support. | Upgrade the process so that it provides equivalent results using a service supported by AEM Forms. (See Aboutdeprecated operations.) |

## WBP-153

`Operation 'operation name' of Service 'service name' used by activity 'activity name' is deprecated.`

This validation message applies to a service operator.

| Explanation | Resolution |
|---|---|
| The activity is configured to use an operation that AEM Forms does not support. | Replace the operation with one that AEM forms supports. (See About deprecatedoperations.) |

## WBP-170

`The variable 'variable_name' is not referenced in the process 'process_name'.`

This is a variable validation message.

| Explanation | Resolution |
|---|---|
| The named variable was created but is not used. | Perform one of the following tasks:<br>• Delete the variable. (See Deletingvariables.)<br>*Reference the variable. For example, use the variable as the value of an operation property.*<br>*Verify that references to the variable use the correct name. For example, referencing the variable named cmroot using the name cmRoot is not correct.* |

## WBP-177

`The event 'event type name' referenced by event catch 'event catch name' is inactive.`

This validation message applies to an event.

| Explanation | Resolution |
|---|---|
| The event type of the event catch is not active. | Perform one of the following tasks:<br>• Activate the event type. (See Activatingand deactivating event types.)<br>*Remove the event catch. (See Deletingevents.)* |

## WBP-178

`The event '`*`event type name`*`' referenced by event receive '`*`event receive name`*`' is inactive.`

This validation message applies to an event.

| Explanation | Resolution |
|---|---|
| The event type of the event receive is not active. | Perform one of the following tasks:<br>• Activate the event type. (See Activatin-gand deactivating event types.)<br>*Remove the event receive. (See Deletin-gevents.)* |

## WBP-179

`The event '`*`event type name`*`' referenced by event startpoint '`*`event start point name`*`' is inactive.`

This validation message applies to an event.

| Explanation | Resolution |
|---|---|
| The event type of the event start point is not active. | Perform one of the following tasks:<br>• Activate the event type. (See Activatin-gand deactivating event types.)<br>*Remove the event start point. (See Delet-ingevents.)* |

## WBP-180

`The event '`*`event type name`*`' referenced by event throw '`*`event throw name`*`' is inactive.`

This validation message applies to an event.

| Explanation | Resolution |
|---|---|
| The event type of the event throw is not active. | Perform one of the following tasks:<br>• Activate the event type. (See Activatin-gand deactivating event types.)<br>*Remove the event throw. (See Deletin-gevents.)* |

## WBP-181

`The event '`*`event type name`*`' referenced by event catch '`*`event catch name`*`' does not exist.`

This validation message applies to an event.

| Explanation | Resolution |
|---|---|
| The event type of the event catch does not exist. The event type was deleted after the event catch was added to the process diagram. | Remove the event catch. (See Deletingevents.) |

## WBP-182

`The event '`*`event type name`*`' referenced by event receive '`*`event receive name`*`' does not exist.`

This validation message applies to an event.

| Explanation | Resolution |
|---|---|
| The event type of the event receive does not exist. The event type was deleted after the event receive was added to the process diagram. | Remove the event receive. (See Deletingevents.) |

## WBP-183

`The event '`*`event type name`*`' referenced by event startpoint '`*`event start point name`*`' does not exist.`

This validation message applies to an event.

| Explanation | Resolution |
|---|---|
| The event type of the event start point does not exist. The event type was deleted after the event start point was added to the process diagram. | Remove the event start point. (See Deletingevents.) |

## WBP-184

`The event '`*`event type name`*`' referenced by event throw '`*`event throw name`*`' does not exist.`

This validation message applies to an event.

| Explanation | Resolution |
|---|---|
| The event type of the throw catch does not exist. The event type was deleted after the event throw was added to the process diagram. | Remove the event throw. (See Deletingevents.) |

### WBV-001

`file name that file name depends on has been updated`

This validation message applies to files.

| Explanation | Resolution |
|---|---|
| Workbench has detected that a file referenced by another file has been updated. The referenced file has a more recent Date Modified value than the file that references it. | • Open the file that references the updated file and determine if changes are required to reflect the updates. |

### WBV-002

`Could not locate dependency: file name`

This validation message applies to files.

| Explanation | Resolution |
|---|---|
| Workbench cannot locate a file referenced by another file. | • Replace the missing file.<br>*Open the file that references the missing file and update or remove the invalid reference.* |

## 16.3. Recording and playing back process

Record process instances as they execute at run time and then play them back to observe the behavior of the process:

- See which routes are followed when the process is executed.
- Determine the values that variables are assigned at each step in the process.

Recording and playing back is useful for testing process versions in the development environment before performing formal testing in a staging environment.

**NOTE:** If your process includes no operations or includes only an event start point, you need to add the Decision Point operation to the process diagram so that process Input variable values are recorded.

Process recordings are saved on the AEM Forms Server so that they are available to other developers running other installations of Workbench.

Before testing process versions by recording and playing them back, you should validate them. (See Validationreports.)

The following procedure describes the tasks that you perform to record process executions and play them back:

1) Enable recording for the process version. (See Enablingand disabling recording.)
2) Record and playback a process when invoking a process. (See Invoking processesdirectly.)
3) Invoke the process version. (See Invokingprocess versions.)
4) Play back the recorded process instance. (See Playingback process recordings.)
5) Delete recorded process instances that you no longer need. (See Deleting recordings.)
6) Disable recording for the process version. (See Enablingand disabling recording.)

**NOTE:** In case an endless loop is executing in a process instance, for each process instance a maximum of 250 operation executions can be recorded. This limitation preserves server resources in the event of an endless loop.

## Enabling and disabling recording

Enable recording for a process version to record the execution history of the process version. While recording is enabled, a new recording is saved each time the process version is invoked. Disable recording when you no longer want to record process executions.

**NOTE:** Recording uses resources on the AEM Forms Server, and reduces performance. You should record only in development environments.

When recording is enabled for a process version, the icon for the process version in the Processes view includes the image of a red recording button in the upper-right corner .

When you create recordings for a process version and then make changes to the process version, the recordings are still available for playback.

To enable recording, the process version must be activated.

Keep in mind that you can also enable record process executions when invoking a process. (See Invokingprocesses directly.)

**To enable or disable recording for a process version:**

1) In the Applications view, right-click the process version and click Record And Playback > Start Recording or Record And Playback > Stop Recording.

   *If a process version has recording enabled, recording is automatically stopped if the process is deactivated.*

   **TIP:** If the process version is open, you can right-click an unused area of the process diagram to access the commands.

## Invoking process versions

When recording is enabled for a process version and you invoke the process version, the execution of the resulting process instance is recorded. You can invoke process versions directly by using Workbench or using the endpoints that are configured for the process service:

- Invoke directly only to test that the process behaves as expected. (See Invokingprocesses directly.)
- Invoke using endpoints to test the use of the endpoints in addition to process behavior. (See Invokingprocesses using endpoints.)

### Invoking processes directly

You can invoke process versions from within Workbench for testing purposes. Invoking directly is useful for focusing only on the process version functionality.

After focusing on the process version, you should perform tests that mimic usage scenarios that are expected in the production environment. (See Testingusing realistic scenarios.)

When you invoke process versions directly, specify values for input variables. The results of the process execution appear in a dialog box. The information that the dialog box provides depends on whether the process version is long-lived or short-lived:

**Long-lived:**

The dialog box indicates the identification of the process instance. (See also Short-livedprocesses and long-lived processes.)

**Short-lived:**

The dialog box indicates the values of the output variables for the process instance.

If you invoke a process version multiple times, you can either use the same values for input variables or specify different values.

Keep in mind that you can optionally record process executions and play them back when invoking a process.

*To invoke a process version directly:*

1) In the Applications view, right-click the process version and select Invoke Process.
2) Specify values for the input variables of the process version. Required values are denoted with an asterisk (*):
   - If you previously invoked the process version directly, the input variables are automatically configured with the previously-used values.
   - To change all existing values, click Reset to remove the values and then specify new values.
3) (Optional) To record process executions, do one of the following:
   - Short-lived process: To record process executions and play them back, select Enable and Playback Recording.
   - Long-lived process: To record process executions, select Enable Recording.

4) Click OK.

**TIP:** If the process version is open, you can right-click an unused area of the process diagram to access the Invoke Process command.

### Invoking processes using endpoints

Invoke process versions to execute them and create process instances. You can invoke process versions by using various tools that AEM Forms provides:

- AEM Forms programming APIs and EJB, SOAP, and remoting end points.
- Watched Folder or Email endpoints.
- Workspace and TaskManager endpoints. (For information about processes that integrate with Workspace, see Designinghuman-centric processes.)

After you activate a process version, EJB, SOAP, and Remoting endpoints are automatically created for the process version's service. You must manually create Watched Folder, Email, and Task Manager endpoints by using Applications and Services.

For more information about creating endpoints, see "Managing Endpoints" in Applications and Services Administration Help.

## Playing back process recordings

Play back recorded process instances to observe the behavior that occurred at run time. When you play a recording, the associated process version is opened in playback mode:

- Routes that were followed at run time are highlighted in green.
- Playback controls enable you to step through the recording.

As you play the recording, you can see the value that process variables contain after each operation was executed. This information is valuable for troubleshooting unexpected behavior.

RELATED LINKS:

Opening recordings in playback mode

Playing recordings

Interpreting recordings

Recording and playing back process

Testing and troubleshooting process versions

### Opening recordings in playback mode

Open recordings in playback mode so that you can see the routes that were followed at run time and to play the recording.

**NOTE:** You cannot edit process maps when they are open in playback mode.

**A.**

Playback controls

**B.**

Route that was followed

To open a recording in playback mode, you select the recording from a list of existing recordings. The list provides the following information so that you can identify the recording that you are interested in:

**Process Instance ID:**

The unique identification of the process instance that was recorded.

**Recording Started:**

The date and time when the recorded process instance was invoked.

**Process Version:**

The number of the process version, and the date and time when the revision was saved.

Recordings are associated with the revision of the process version that they are made with. When you open a recording that was created with a prior revision of a process version, the process diagram appears as it did for that revision.

For example, after you create a recording for a process version, you add an operation to the process map and save the changes. The recording that you created is associated with the previous revision of the process version. When you open the process recording in playback mode, the process diagram appears without the operation that you saved in the latest revision.

**TIP:** You can specify that recordings for a process version are deleted automatically when the process version is saved. (See Customizingrecord and playback behavior.)

You need to be assigned the Process Administrator role or have the Process Recording Read/Delete permission to play and delete recordings. Contact your AEM Forms administrator, or use administration console to set these permissions.

*To open a recording in playback mode:*

1) In the Applications view, right-click the process version to play a recording for and select Record And Playback > Play Process Recording.
2) Select the recording to play and then click OK.

**TIP:** If the process version is open, you can right-click an unused area of the process diagram to access the Record And Playback commands.

RELATED LINKS:

Playing recordings

Recording and playing back process

Testing and troubleshooting process versions

### Playing recordings

Play recorded process instances to observe process behavior at run time. When playing, you can see variable values at each operation, and information about any errors that have occurred. (See Interpretingrecordings.)

**NOTE:** Before you play a recording, you need to open the recording in playback mode. (See Openingrecordings in playback mode.)

When playing recordings, the progress line indicates the current state of the process instance during execution:

- Ticks on the progress line represent the operations that executed at run time.
- The direction of progress is from left to right. The first tick on the progress line represents the process before it is executed.
- The slider on the progress line indicates the current state of progression. The tick where the slider is positioned represents the last operation that completed executing. This operation is also highlighted on the process diagram.

For example, a progress line that includes five ticks indicates that four operations executed at run time. If the slider is located at the third tick during playback, the second of the four operations has completed executing.

You can either play the entire recording or manually step through the recording. Manually stepping through the recording allows more time for observing variable values at each operation.

**TIP:** You can specify the amount of time that playback pauses at each tick. (See Customizingrecord and playback behavior.*)*

*To play the entire recording:*

1) Click the play button .
   *While playing, the pause button appears instead of the play button.*
2) To pause playback at a tick, click the pause button .

*To manually step through the recording:*

1) To advance to the next tick, click the step forward button or click the next tick on the progress line.
2) To move to the previous tick, click the step backward button or click the previous tick on the progress line.

RELATED LINKS:

Interpreting recordings

Deleting recordings

Recording and playing back process

Testing and troubleshooting process versions

### Interpreting recordings

Visual indicators on the process diagram provide information about the results of the process execution:

- Routes that were followed during execution are highlighted in green. Routes that were not followed are blue.
- As you play or manually step through the recording, the last operation that completed executing is highlighted. The color of the operation indicates the success of its execution:
  - Green indicates that the operation executed successfully.
- Red indicates that an error occurred during execution.

Use the Variables view to see the run-time values of variables:

- The values correspond with the current state of execution during playback.
- Variables whose values changed are highlighted in green. Variables are highlighted only at the steps that cause a change to occur.
- Values of simple data types appear on the view. Values of complex data types are displayed in a separate dialog box.
- If a variable value is a document, you can open the document in the application that it is associated with. For example, if your operating system associates XML files with a web browser, then XML files open in the web browser.
- If a variable is used to store form data and the form location, you can open the form and see the form data.

The Variables view also shows information about any errors that occurred during execution. Errors in the view are highlighted in red. The columns contain the following information:

**Name:**

Contains `Error`.

**Type:**

The type of error and the error message.

**Value:**

An ellipsis button that opens an Error dialog box that shows the error message and stack trace. Use this information to determine the cause of the error.

If no errors occurred during execution, no Error row appears in the Variables view.

*To see variable values during playback:*

1) When the recording is playing, see the value in the Value column of the Variables view. For complex data types, documents, lists, and maps, click the ellipsis button to see the data or open the document .

   **TIP:** Right-click a variable and select Copy to copy the value of the variable. You can paste the value in an email message for communicating with other developers.

*To see details about errors:*

1) In the Error row on the Variables view, click the ellipsis button in the Value column .
2) Click Details to see the stack trace.

RELATED LINKS:

Playing recordings

Recording and playing back process

Testing and troubleshooting process versions

## Deleting recordings

You can delete recordings at any time when you no longer need them.

You need to be assigned the Process Administrator role or have the Process Recording Read/Delete permission to delete recordings. Contact your AEM Forms administrator, or use administration console to set these permissions.

**TIP:** You can specify that recordings for a process version are deleted automatically when the process version is saved or deleted. (See Customizingrecord and playback behavior.)

**To delete a recording:**

1) In the Applications view, right-click the process version for which you want to play a recording and select Record And Playback > Delete Process Recordings.
2) Specify the recordings to delete:
   – To delete all recordings, select Delete All Recordings.
   – To delete one or several recordings, select Delete The Recordings Selected Below, and select the recordings to delete.
3) Click OK.

RELATED LINKS:

Limiting storage space used for recordings

## Customizing record and playback behavior

Several preferences can be configured that control some of the behavior of the record and playback features. The following table describes the preferences that you can configure.

| Preference | Default value | Description |
|---|---|---|
| -Dcom.adobe.workbench.recordings.remove_on_save | `false` | Controls whether recordings for a process version are removed when the process version is saved.<br>A value of `true` keeps recordings that are associated with prior revisions of process versions. |
| -Dcom.adobe.workbench.recordings.remove_on_delete | `true` | Controls whether recordings for a process version are removed when the process version is deleted.<br>A value of `true` removes all recordings for a process version when it is deleted. |
| -Dcom.adobe.workbench.playback.max_recordings_to_display | `100` | The maximum number of recordings that are listed when selecting recordings to delete or play. |
| -Dcom.adobe.workbench.playback.animation.milliseconds_to_delay | `2000` | The amount of time, in milliseconds, that the slider pauses at a tick on the progress line when playing recordings. |
| -Dcom.adobe.workbench.unsupported.audit.maxNumberOfStepsToLoad | 250 | The maximum number of steps for each recording that can play back. |

To configure preferences, modify the workbench.ini file. Each line in the file contains a preference setting in the following format:

```
preference_name = value
```

For example, the following preference setting specifies that recordings for prior revisions of process versions are saved:

```
-Dcom.adobe.workbench.recordings.remove_on_save=false
```

The workbench.ini file is located in the *[install directory]*/AEM forms Workbench/Workbench directory, where *[install directory]* is the location where you installed Workbench.

**NOTE:** If the -Dcom.adobe.workbench.recordings.remove_on_delete preference is set to `true`, but you remove a process version by deleting the process archive in the administration console, the recordings for that process are not removed. To remove the recordings, stop and start the Audit Workflow DSC component. (See Stopping componentsand servicesand Startingcomponents and services.) This action will remove all the recordings you created.

**To customize record and playback behavior:**

1) Open the workbench.ini file in a text editor.
2) Modify the values of preferences as required.
3) Save workbench.ini and restart Workbench.

## Limiting storage space used for recordings

To conserve space on the AEM Forms Server's file system, you can limit the amount of process recording data that is stored. You can configure the following properties of the AuditWorkflowService service:

**maxNumberOfRecordingInstances:**

The maximum number of recordings that are stored. When the maximum number is stored, the oldest recording is removed from the file system when a new recording is created. This property is useful if you tend to create many recordings and you want to remove old recordings automatically.

**maxNumberOfRecordingEntries:**

The maximum number of data entries that can be stored for each recording. Data entries are information about operations in the process. Several entries are stored for each execution of an operation, such as whether the operation started, whether the operation completed, and whether the route that leads to the operation is complete. This property is useful when processes can include many operation executions, for example, when an endless loop is encountered.

The AuditWorkflowDSC component provides the AuditWorkflowService service. Use the Components view to configure the service properties. (See Editing serviceconfigurations.)

Processes can also be designed to minimize the storage space used for recordings:

- Use counters in loops to limit the number of loops that are executed. (See Loops.)
- Segment your processes using subprocesses and enable recording only for the subprocesses that you are testing. (See Processdesigns for reuse.)

### Configuring server memory for large recordings

Playing recordings of processes that include many variables or many operations can require the use of large amounts of memory on the AEM Forms Server. When the AEM Forms Server becomes destabilized when you try to play a process recording, the Java Virtual Machine (JVM) that hosts the application server can require more memory than it is allocated. Also, you may need to restart the computer that hosts the server to establish stability.

To increase the memory that is allocated to the JVM, configure the -Xmx option of the JVM that runs the application server. For example, if the current option is `-Xmx1024m`, change the option to `-Xmx1280m` to increase the maximum amount of allocated memory by 256 MB.

The following table lists links to AEM Forms documentation that provide information about how to configure JVM options on your application server.

| Application server | Documentation |
|---|---|
| JBoss | See the “Modifying the JBoss configuration files” topic in the Preparing To Install AEM Forms guide. |
| WebSphere | See the “Configuring the JVM arguments and properties” topic in the Installing and Deploying AEM Forms for WebSphere guide. |
| WebLogic | See the “Configuring the JVM arguments” topic in the Installing and Deploying AEM Forms for WebLogic guide. |

## 16.4. Testing using realistic scenarios

When you test process versions, use realistic test scenarios that mimic the environment and use cases that are expected in the staging and production environment. Realistic scenarios help to reduce issues found during staging and production testing. It is recommended that you look at these settings in your development environment:

**Platforms:**

Develop and test on the same platform as the production environment when possible. For example, prepare and perform your development testing on the same application server and database that the production environment will use. You also need to consider using the same Sun Java™ JDK versions.

**Operating system and web browsers:**

Prepare and test on the same operating system when possible. Also, use the same web browsers to perform your testing and consider using same the patch levels for both the operating system and web browser.

**Users, groups, and roles:**

Test with user profiles, group profiles, and roles that have similar permissions and system privileges as in the production environment. For many development environments, developers have

extended privileges that a typical user does not have in a production environment. For this reason, you need to test with similar user profiles and group profiles as found in the production environment. Configure user profiles, group profiles, and roles in administration console. For information about configuring users, see *User Management Help,*accessible in the administration console.

**Location of assets:**

Test with an identical directory or folder structure for any assets used by the process version you design in your production environment. Assets include forms, images, fragments, SWF files, and data files you design. For example, in a production environment, the form can be located at a URL instead of a specific folder structure in the repository.

**Acrobat and Adobe Reader:**

It is recommended that you test with the same editions, versions, and patch levels of Acrobat or Adobe Reader that an end user will use in the production environment. For example, if you production environment only uses Adobe Reader, it is recommended that you to use Adobe Reader for testing.

**Adobe Flash Player:**

It is recommend that you test with the most current supported version of Flash Player that is used in the production environment.

**NOTE:** Some versions of Acrobat, Adobe Reader, and Flash Player may not be supported in your production environment.

## 16.5. Test cases to consider

Consider the following test cases when you perform development testing on your process version:

**Input:**

Your process version may require specific input. It is recommended when you test that you use a mix of valid and invalid inputs. For example, your process version may require a number to be entered as input. You can test exception handling if the number is out of a certain range.

**Output:**

If your process version provides an output that can be used by another process that another process can use, you may want to consider testing the output by using the service as a subprocess of another process.

**Output files:**

If your process version produces ancillary files, you may want to confirm that they are generated correctly. For example, your process version may encrypt a PDF file and save it to a location on the disk. You may want to test that the file is correctly saved to the specified location.

**Business logic:**

You may have a process version where different routes can be executed. Consider testing all possible business logic:

- Use a range of valid and invalid values at the decision points in your process version.
- If you use values stored in variables for decision making in the process version, consider using the Variable Logger service.
- Validate that events in your process version throw and receive notifications when specific business logic is executed.

**Users in a process:**

For human-centric processes, a user can invoke a service or participate in a process using Workspace. Consider these specific use cases for human-centric processes:

- Test the forms that are used with your process version. This includes tasks such as putting a mix of valid and invalid values into the form.
- If notes and attachments are configured, verify that users can use them.
- Test the restrictions on delegating and consulting if they are configured.
- Test time constraints that you have configured in the process version, which include deadlines, reminders, and escalation.

**Events:**

Your process version may include events that act as start points to invoke your service. You may want to test that your service is invoked correctly. Another example would be for event receives and event throws that are used in your process version. You may want to verify that an event throw is sending the correct information and that an event receive properly filtered the information.

# 16.6. Handling errors in the production environment

You can design processes so that they gracefully handle errors when they occur:

- Use the Variable Logger service to monitor process data at run time.
- Use event catches to handle execution errors.
- Use the Stall service to suspend execution when undesirable situations occur.

## Execution errors

An execution error causes a process instance to stall. A process instance fails to complete processing for one of these reasons:

### Stalled operation errors

A stalled operation error occurs when an operation in your process version fails during execution. For example, the execute operation from the Set Value service tries to assign a string value to a process variable of the float data type. This value is not valid and causes the operation to stall.

### Stalled branch errors

A stalled branch error occurs when a routing condition has an error. For example, you may have a condition on a route that uses a value from a form. The route condition is based on a numeric value, but the form value is inadvertently entered as a string value. This error causes the branch to stall.

You can view the list and details of stalled operation errors or stalled branch errors from within administration console. You will see the name of the operation or branch that is stalled, and you can view the details of the trace back that may provide useful information to identify the source of the problem.

## Implementation errors

Implementation errors occur when a process version completes execution successfully, but it does not work as it was designed. For example, if you have a route condition to route all high-risk loan applications to the manager, and the manager never gets routed any of these loan applications, you have an error in your process version. The two types of implementation errors are computational and situational.

### Computational errors

Computational errors occur when the execution of a command results in an error. For example, an error can occur in a routing condition when the function in an expression uses a parameter of the wrong data type. When the error occurs, the branch or action involved is stalled until the AEM Forms administrator intervenes. The type of branch that your process version uses can mitigate the effects of computational errors.

### Situational errors

Situational errors occur when an issue that should not occur does occur during the execution of a process version, even though the process version behaved as designed. For example, a process version assigns a task to a random user in a specific user group. If that user does not complete the task by a certain time, it is incorrectly reassigned to the same user in the group again.

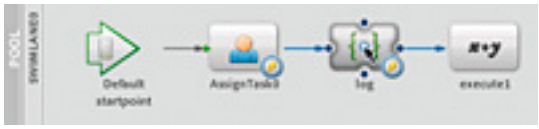
A situational error occurs if the task was reassigned to the same person. To avoid situational errors, you can cause the branch to stall before the situation occurs. The administrator can then intervene and attempt to correct the situation.

## Monitoring variables using the Variable Logger service

Use the log operation that the Variable Logger service provides to monitor the values of process variables at run time in a production environment. The Log operation writes variable values to a file, to standard out, or to the AEM Forms Server log at a specific point in the process.

**TIP:** For information about viewing variable values during testing, see Recordingand playing back process.

For example, the following process diagram includes a log operation after an Assign Task operation. After the log operation, an execute operation from the Set Value service manipulates the data that the Assign Task operation captures. The log operation saves variable data to a file. If the execute operation stalls at run time, the logged variable values can be used to troubleshoot the error.




The log operation reports the following information for each variable in the process:

- The name of the variable, as an XPath expression
- The data type of the variable
- The value of the variable
- The process identification (PID) of the process instance in which the log operation executed.
- The operation identification (ActionID) of the log operation.

The following text is an example log entry that a log operation named log3 provides. The log3 operation has an ActionID of 37, and was executed in a process instance with PID 13. The process includes the two variables named numberVar and listVar:

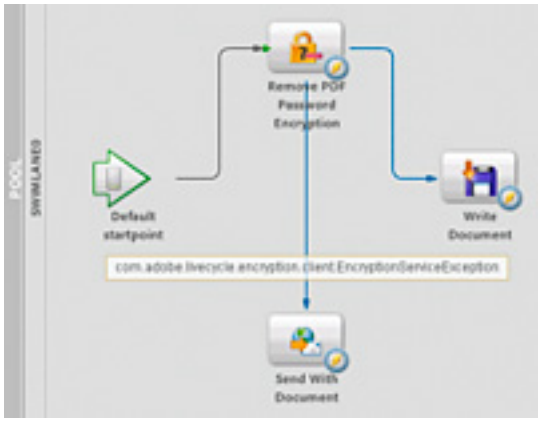
```
[PID:13] [ActionID:37] Action Name: "log3" Start...
[PID:13] /process_data/@numberVar - java.lang.Float: 100000.0
[PID:13] /process_data/@listVar: null
[PID:13] [ActionID:37] End!
```

For more information about the log operation, see log.

## Handling errors using exception event catches

Exception event catches enable processes to react to errors that occur when an operation executes. For example, when an exception event is caught, the process can write information to a log file, or send an email to the AEM Forms administrator. (See also Catchingevent throws.)

The following process diagram uses a Remove PDF Password Encryption operation to decrypt a PDF document, and then saves the document to the hard drive using the Write Document operation. If an EncyrptionServiceException exception occurs when the document is being decrypted, the Send With Document operation sends an email to notify the administrator.



### Handling situational errors using the Stall service

Use the execute operation that the Stall service provides to prevent situational errors that you anticipate.

For example, processes can use data that is provided from an external resource, such as a partner's database. The executeScript operation from the ExecuteScript service can be used to verify that the data is valid. If the data is not valid, the Execute operation from the Stall service can stall the process instance while the data in the database is corrected.

RELATED LINKS:

Implementation errors

Handling errors in the production environment

Process execution

Process life cycle

Testing and troubleshooting process versions

## 16.7. Inspecting application server logs

You can inspect the application server logs if you have access to the AEM Forms Server. Logs are useful for looking at Java exception tracebacks to provide information on processes that may have stalled or have encountered errors. If you used the Variable Logger service in your process, you can also view the logs to see variable values that are written to application server logs.

Depending on the application server and how you installed AEM forms will determine the location of the server logs. For example, in a JBoss turnkey installation, the application server logs are located at*[install directory]*\Adobe\AEM Forms\jboss\server\all\logs where*[install directory]* represents the location where AEM forms was installed. For information about application server logs, see Administering AEM Forms.

You can also install the server log viewer to inspect the application server logs without access to the application server.

### Edit server log viewer configuration settings

1) In the Components view, right-click ServerLogComponent > Active Services > ServerLogService and select Stop Service
2) Right-click ServerLogComponent > Active Services > ServerLogService and select Edit Service Configuration, and modify the following settings as required:

- **serverLogFolder**: Specifies the location of application server log. The default value is C:/Adobe/AEM Forms/jboss/server/lc_turnkey/log for a JBoss application server.
- **serverLogName**: Specifies the name of the server log file. The default value is server.log
- **batchSize**: Specifies the number of log messages to retrieve from the server log file. Each line in the log is a message. The default value is -1, which indicates to retrieve all of the messages in the log file.

3) Right-click ServerLogComponent > Active Services > ServerLogService and select Start Service.

RELATED LINKS:

Handling errors in the production environment

Monitoring variables using the Variable Logger service

# 17. Managing event types

Event types are templates for events. As well as using the event types provided with Workbench, you can create your own event types for any purpose you require. You can also organize event types into predefined categories or groups.

For information about using events, see Controllingprocess flow using events.

Furthermore, you can import and export event types between servers, and you can activate and deactivate event types for management or maintenance.

## 17.1. Event type properties

Event types have the following properties:

**Event Category:**

The category to which the event type belongs. The event categories are Asynchronous, Exception, and Timer. You can create custom Asynchronous and Timer events.

The following tabs contain additional properties of the event type:

**Data Schema:**

Specifies the XML Schema Definition (XSD) of the data elements for the event data. The XSD describes the data elements reported on the event, the data types of the elements, and the order and multiplicity of each element.

**Message Schema:**

Specifies the XSD of the data elements for the event message data. The XSD describes the data elements for asynchronous event types, the data types of the elements, and the order and multiplicity of each element.

**Configuration:**

Specifies how the settings for the event type are configured and include the following properties:

- **Event Priority**: Specifies the order in which processes handle simultaneous events. Valid values for priority are from 1 to 9, with 9 being the highest priority.
- **Event Severity:** Sets how severe the AEM Forms Server considers the event. Severity does not necessarily drive the priority. Severity is one of LOW, MEDIUM, or HIGH. In most cases, a HIGH severity event requires you to use high priority handling.
- **Initial States:** Indicates whether the event type is initially Active or Inactive. An active event type indicates that events of that type can be thrown or received. If the event type is inactive, events of that type are not thrown or received. When the event type is reactivated, events are thrown or received again from that time forward

- **Event Activation:** Indicates whether the event type is deactivated Upon Notification or Never. Upon Notification deactivates the event type when the listeners have acknowledged receipt of the event. Never specifies that the event type is never deactivated.

### Display the properties of event types

To display the properties of an event type, complete one of the following tasks:

- If the Events view is not visible, select Window > Show View > Events.

  *In the Events view, double-click an event.*
- In the process diagram, right-click the event, and select Event Properties.
- In the process diagram, select the event, and in the Process Properties view, click Event Properties.

RELATED LINKS:

Creatingcustom event types

Organizingevent types

Exportingand importing event types

Activatingand deactivating event types

## 17.2. Custom event types

If the provided event types are not suitable for a specific purpose in a process, you can create your own custom event type. You can create custom Asynchronous and Timer event types that can be used in the same ways as other event types. You cannot create custom Exception events.

Another process can listen for that event, filter it based on order quantity, and then trigger a particular behavior. For instance, the other process can send an email or route a form or data to an inventory clerk or a production manager.

For instance, suppose you want to provide a mechanism to inform a process by way of an event that a purchase order has been created. To do this, you can create a PurchaseOrder event type that contains the purchase order number, the name of the company placing the order, the part number being ordered, and the quantity of the order. You then create a process that throws this event type when the order occurs.

Creating subfolders in the application tree is a recommended way of organizing your assets inside the application. For example, create a folder named customEvents to store custom events.

When you create a custom event type, the combined length of the Name field and the Enter Or Select The Parent Folder field must not be greater than 260 characters. For example, if the name of the event is `SampleEventTypeDemo`, and the parent folder path is `/SampleDemo-AssemblerService/1.0/CustomEvents/`, the combined name is: `/SampleDemo-AssemblerService/1.0/CustomEvents/SampleEventTypeDemo`. The combined length is 65, and is an allowed name.

## Create custom event types

1) Select File > New > Event Type. You can select an event that is located in the current application or in another application that is either local or remote.
2) In the New Event Type dialog box, in the Name box, type a name for the event type that is meaningful and descriptive to other users.
3) (Optional) In the Description box, type a description of what the event type is used for.
4) (Optional) Create a subfolder by completing the following steps:
   – Below the application tree area, click New Folder.
   – In the New Folder dialog box, in the Folder Name box, type a name for the folder and click OK.
5) In the Enter Or Select The Parent Folder box, specify the path to the event location within the application hierarchy. The path must include the name and version of the application and, optionally, subfolder names. You can either type the path or select the required location from the application tree. The application you choose can either be local or remote. An example of a valid path is /MyApp/1.0/customEvents.
6) Click Next.
7) Under the Event Category label, select the Asynchronous or Timer option. (See Eventcategories.)
8) Import an XML Schema Definition (XSD) file to use for event data by completing these steps: (See Creatingdata schemas for event data and messages.):
   – In the Data Schema tab, click Import Schema.
   – In the Data Schema dialog box, select the location of the XSD file in the application tree, and then click OK.
9) Import an XSD file to use for additional event message data by completing these steps: (See Creatingdata schemas for event data and messages.):
   – In the Message Schema tab, click Import Schema.
   – In the Message Schema dialog box, select the location of the XSD file in the application tree, and then click OK.
10) In the Configuration tab, configure the custom event type by completing these steps:
   – Below Event Priority, select the priority level of the event type. The value of 1 represents the lowest priority and 10 represents the highest priority.
   – Below Event Severity, select LOW, MEDIUM, or HIGH to set the severity of the event.
   – Below Event State, select Active or Inactive. Select Active if you want the event type to be active when it is first created.
   – Below Event Activation, select Upon Notification or Never. Upon Notification specifies that the event is deactivated after it is received.
11) Click Finish.

### Delete custom event types

Do not delete an event type that is being used in any process diagram. If you try to delete such an event type, an error dialog box appears. If you are not using a custom event type, you can delete it.

1) In the Applications view, right-click the Event from an Application version,
   *and select Delete.*

RELATED LINKS:

Displayingthe properties of event types

Organizingevent types

Exportingand importing event types

Activatingand deactivating event types

Creatingevent filters

## 17.3. Creating data schemas for event data and event message data

When you create an event type, you specify an XML Schema Definition (XSD) file as data schema for event data and for event message data. Data schemas define the format of the data and data elements. These XSD files define the data structure displayed as navigation trees t in the event configuration dialog boxes. The data structure determines the elements you can use to filter events and map to values to when events are thrown or received.

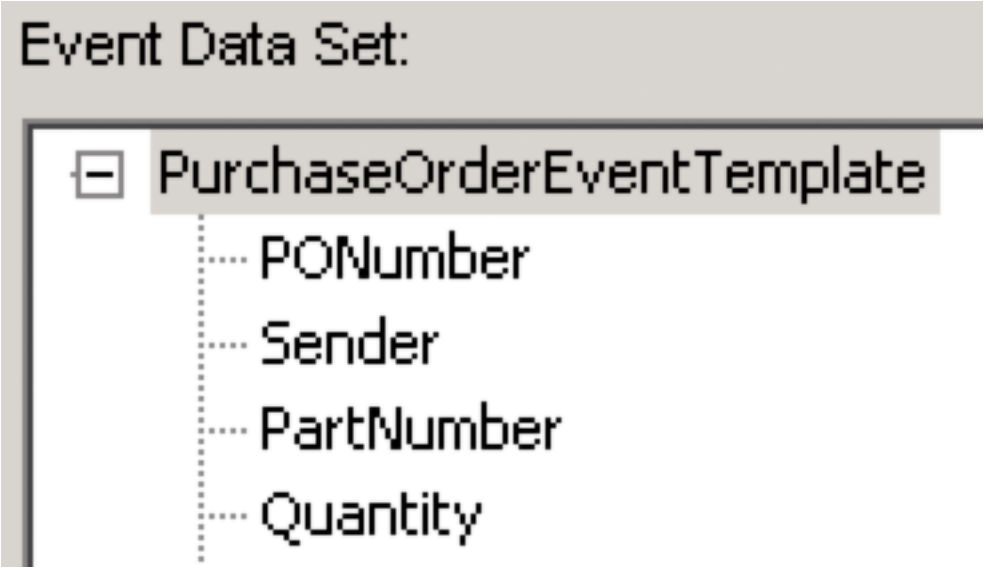
You create or modify a data schema using an XML or text editor and then import it as an asset to an application. You can also use a similar existing data schema from another event type. The data schema must adhere to standard XML syntax rules.

When you create an event type, you can import data schemas for the event data and event message data. The data schema is available immediately after you save the event type and the data trees are visible in the event configuration dialog boxes.

For example, consider the following partial data schema for event data:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="unqualified" attributeFormDefault="unqualified">
<xs:element name="PurchaseOrderEventTemplate">
<xs:complexType>
<xs:sequence>
<xs:element name="PONumber" type="xs:short"/>
<xs:element name="Sender" type="xs:string"/>
<xs:element name="PartNumber" type="xs:long"/>
<xs:element name="Quantity" type="xs:long"/>
...
</xs:sequence>
 </xs:complexType>
</xs:element>
</xs:schema>
```

That data schema results in the following data navigation tree in all related event configuration dialog boxes.




## Data schemas for event data

To specify the data elements, create a data schema for the event data. The data schema is useful when you filter event data from event receives. The data schema can also be used to define the data that is exchanged between processes. For ease of use, a simple structure with a small set of data elements is recommended.

To throw an event, you assign values to the data elements by specifying the XPath in the XML Schema Definition (XSD). Any such data assignment must conform to the XSD restrictions. For example, if an element is specified as a mandatory element in the XSD, the value must be provided. Similarly, if an element has length restrictions in the XSD, that length restriction must be adhered to.

To filter data, an event receive or event catch can use elements in the XSD for event data to compare values with other data.

**NOTE:** The XML representing the event data that an event throw generates does not support namespaces.

## Data schemas for event message data

Data schemas for event message data only apply to asynchronous events and are optional. The XSD for these data schemas identify the event message data that an asynchronous event receives or throws.

The XSD for the event message data can be a relatively complex data structure, such as the data definition for a purchase order or sales order. The event message data is not generated by a process. Instead, the event message data is assigned to the event when it is thrown from a process variable of type `xml`. The event message data can be copied as a whole into the process variable. Alternatively, specific data from the `xml` variable can be introduced into the process. Event filters can be specified on the elements specified in the data schema for event receives.

In contrast to event data, the event message data does support namespaces. Register the namespace in the Process Properties dialog box to use namespaces. (See Creatingprocesses using the New Process wizard.)

RELATED LINKS:

Creatingcustom event types

Displayingthe properties of event types

Organizingevent types

Exportingand importing event types

Activatingand deactivating event types

Creatingevent filters

# 17.4. Organizing event types

Event types are organized into categories and groups in the Events view.

**Event categories:**

The event categories are Asynchronous, Exception, and Timer. You cannot create other event categories.

**Event groups:**

Workbench 9.5 has a set of predefined event groups. You cannot create other event groups. The event types in the group are called *event group members* and are proxies to the event types. In Workbench, the primary use of event groups is for importing, exporting, activating, or deactivating a set of event types. To see Custom event types in the Events view, click Refresh after you deploy your application.

## Displaying event types in categories or groups

1) In the Events view, click Display By Event Group or Display by Event Category.

   **NOTE:** The same button is used to toggle between categories and groups.

# 17.5. Exporting and importing event types

If you have an event type on a server that you want to use on another server, you can export the custom event types from that server and then import it into the other server. The exported file is an XML file.

You can export custom events when you are displaying events by category.

**NOTE:** To export more than just event types, create an archive file instead. An archive includes the process diagram, operations, and other resources. When you create an archive, event types are included but not the event groups. (See Aboutarchive files.)

### Export event types

1) In the Events view while in the event category view, right-click an event type and select Export.
2) In the Export dialog box, specify the location of the exported file, and click Save.

### Import event types

1) Select File > Import.
2) In the directory tree, expand General, and then select File System.
3) Click Next.
4) Beside the From Directory box, click Browse.
5) In the Import From Directory dialog box, select the folder where the event type file exists, and click OK.
6) In the right pane, ensure that the folder that contains the file is selected.
7) In the left pane, select one or more event type files to import.
8) Beside the Into Folder box, click Browse.
9) In the Import Into Folder dialog box, select the application folder to import the event type file into, and click OK.
10) Click Finish.

RELATED LINKS:

Displayingthe properties of event types

Creatingcustom event types

Organizingevent types

Activatingand deactivating event types

## 17.6. Activating and deactivating event types

To be used in a process, an event type must be active. To prevent an event type from being used for maintenance or other administrative reasons, you can deactivate it. Deactivated event types are indicated with a universal No symbol in the lower-right corner. You can activate or deactivate a single event type or an event group. When you deactivate an event type that is in use in a process, events of that type cannot be thrown or received, and the process can stall.

To use the event types that Rights Management provides, it is necessary to enable the Rights Management event handlers on the AEM Forms Server. They are disabled by default to conserve server resources at run time.

### Deactivate event types

1) If the Events view is not visible, select Window > Show View > Events.
2) (Optional) In the Events view, click Display By Group or Display By Category button.

3) In the Events view, right-click an event type or event group and select Deactivate. You cannot deactivate individual event types when you display events by group.

### Activate event types

1) If the Events view is not visible, select Window > Show View > Events.
2) (Optional) In the Events view, click Display By Group or Display By Category button.
3) In the Events view, right-click a deactivated event type or event group and select Activate. You cannot activate individual event types when you display events by group.

### Enable Rights Management event types

1) Start administration console by typing the following URL in a web browser:
   `http://[host]:[port]/adminui`
2) In the User ID box, type the administrator user name, and in the Password box, type the associated password. For the specific values to type, see your administrator.
3) After logging in, click Services and then click Rights Management.
4) Click Configuration, and then click Manual Configuration.
5) Click Export, and specify a location to save the configuration XML file.
6) Open the file in a text editor, and locate the SDK element:
   `preferences/system/Adobe/AEM Forms/Config/PolicyServer/SDK`
7) Modify the entry element that has the key attribute with the value EventHandlersEnabled so that the value attribute has a value of true:
   `<entry key="EventHandlersEnabled" value="true" />`
8) Save the file.
9) On the Manual Configuration page of administration console, import the configuration file:
   – Click Browse.
   – Specify the configuration file that you modified.
   – Click Import.
   – Click OK

RELATED LINKS:

Displayingthe properties of event types

Creatingcustom event types

Organizingevent types

Exportingand importing event types

## 17.7. Event type reference

This section provides basic information about the default event types supplied with Workbench.

You can use these event types as supplied or create custom event types using the default event types as a starting point.

## Asynchronous event types

This category of event types includes events related to content services, rights management, task management, and task attachments.

### Asynchronous event type summary

*Content Services event types*

The Content Services event types are thrown automatically when any content in Content Services repository is created, updated, or deleted.

Content Services provides the following event types:

- ContentCreated
- ContentDeleted
- ContentUpdated.

*Content Services review and commenting event types*

In AEM Forms, a user can initiate Acrobat-based shared reviews on documents present in Content Services repository. The review and commenting event types can be used in the process to initiate actions related to shared reviews, such as starting a review, ending a review, and assigning a review.

Content Services provides the following review and commenting event type:

- ReviewAssigned
- StartReviewStage
- TerminateReview.

*Rights Management event types*

The Rights Management service relates to document policy protection. The Rights Management event types are used to capture or track information such as the creation of a new administrator account or policy, closing of a policy-protected document, or the synchronization of the user directory.

Rights Management provides the following event types:

- RMAdminEvent
- RMDocumentEvent
- RMErrorEvent
- RMPolicyEvent
- RMPolicySetEvent
- RMServerEvent
- RMUserEvent.

**NOTE:** To use these event types, you need to enable the Rights Management event handlers on the AEM Forms Server (See Activatingand deactivating event types.)

*Task management event types*

Tasks are usually user steps in a workflow. Tasks can be associated with a process instance or may exist on their own. The task management event types are used to capture or track information when a notable occurrence affects a task, such as a task being canceled, claimed, completed, consulted, created, deadlined, escalated, forwarded, locked, reassigned, rejected, terminated, or unlocked.

forms workflow provides the following task management event types:

- TaskCancelled
- TaskClaimed
- TaskCompleted
- TaskCompletedWithData
- TaskConsulted
- TaskCreated
- TaskDeadlined
- TaskEscalated
- TaskFormDataSaved
- TaskForwarded
- TaskLocked
- TaskReassigned
- TaskRejected
- TaskReminderSent
- TaskShared
- TaskTerminated
- TaskUnlocked
- TaskVisibilityChange.

*Task attachment event types*

Tasks often carry attachments that contain form data, for example. The task attachment event types are used to capture or track information when a notable occurrence affects a task attachment, such as an attachment being added, deleted, modified, or updated.

forms workflow provides the following task attachment event types:

- TaskAttachmentAdded
- TaskAttachmentDeleted
- TaskAttachmentInfoUpdated
- TaskAttachmentUpdated.

### RMAdminEvent

Rights Management administration event related to administrative actions.

**NOTE:** To use this event type, you need to enable the Rights Management event handlers on the AEM Forms Server. (See Activatingand deactivating event types.)

**Priority:**

3

**Severity:**

Medium

**Event Data:**

Audit event data for a single event.

**EventID:**

A `string` value identifying the unique ID associated with the audit in the Rights Management service.

**EventNamespace:**

A `string` value identifying the namespace qualifier for the event type.

**EventType:**

A `string` value representing unique values within an event namespace.

**EventDate:**

A `dateTime` value representing the time when the event occurred.

**EventDetail:**

A `string` value representing the message string containing the event information.

**PrincipalName:**

A `string` value representing the name of the authenticated principal that was used to cause the event to occur.

**PrincipalUID:**

A `string` value representing the unique ID of the authenticated principal that was used to cause the event to occur.

**WasAllowed:**

A `boolean` value representing whether the action was successful or failed.

**WasOnline:**

A `boolean` value representing whether the operation happened online or offline.

**ClientIPAddress:**

A `string` value representing the IP address of the client that connects to the server.

**LicenseID:**

A `string` value representing the unique ID of the license associated with the secured document in the Rights Management service.

**PolicyID:**

A `string` value representing the unique ID for the policy in the Rights Management service.

**Deactivation:**

Upon notification

**Event Message:**

None

**RMDocumentEvent**

Rights Management document event related to opening, securing, and printing.

**NOTE:** To use this event type, you need to enable the Rights Management event handlers on the AEM Forms Server. (See Activatingand deactivating event types.)

**Priority:**

3

**Severity:**

Medium

**Event Data:**

Audit event data for a single event.

**EventID:**

A `string` value identifying the unique ID associated with the audit in the Rights Management service.

**EventNamespace:**

A `string` value identifying the namespace qualifier for the event type.

**EventType:**

A `string` value representing unique values within an event namespace.

**EventDate:**

A `dateTime` value representing the time when the event occurred.

**EventDetail:**

A `string` value representing the message string containing the event information.

**PrincipalName:**

A `string` value representing the name of the authenticated principal that was used to cause the event to occur.

**PrincipalUID:**

A `string` value representing the unique ID of the authenticated principal that was used to cause the event to occur.

**WasAllowed:**

A `boolean` value representing whether the action was successful or failed.

**WasOnline:**

A `boolean` value representing whether the operation happened online or offline.

**ClientIPAddress:**

A `string` value representing the IP address of the client that connects to the server.

**LicenseID:**

A `string` value representing the unique ID of the license associated with the secured document in the Rights Management service.

**PolicyID:**

A `string` value representing the unique ID for the policy in the Rights Management service.

**Deactivation:**

Upon notification

**Event Message:**

None

**RMErrorEvent**

Rights Management error event.

**NOTE:** To use this event type, you need to enable the Rights Management event handlers on the AEM Forms Server. (See Activatingand deactivating event types.)

**Priority:**

3

**Severity:**

Medium

**Event Data:**

Audit event data for a single event.

**EventID:**

A `string` value identifying the unique ID associated with the audit in the Rights Management service.

**EventNamespace:**

A `string` value identifying the namespace qualifier for the event type.

**EventType:**

A `string` value representing unique values within an event namespace.

**EventDate:**

A `dateTime` value representing the time when the event occurred.

**EventDetail:**

A `string` value representing the message string containing the event information.

**PrincipalName:**

A `string` value representing the name of the authenticated principal that was used to cause the event to occur.

**PrincipalUID:**

A `string` value representing the unique ID of the authenticated principal that was used to cause the event to occur.

**WasAllowed:**

A `boolean` value representing whether the action was successful or failed.

**WasOnline:**

A `boolean` value representing whether the operation happened online or offline.

**ClientIPAddress:**

A `string` value representing the IP address of the client that connects to the server.

**LicenseID:**

A `string` value representing the unique ID of the license associated with the secured document in the Rights Management service.

**PolicyID:**

A `string` value representing the unique ID for the policy in the Rights Management service.

**Deactivation:**

Upon notification

**Event Message:**

None

**RMPolicyEvent**

Rights Management policy event related to creation, deletion, activation, or change of policy.

**NOTE:** To use this event type, you need to enable the Rights Management event handlers on the AEM Forms Server. (See Activatingand deactivating event types.)

**Priority:**

3

**Severity:**

Medium

**Event Data**: Audit event data for a single event.

**EventID:**

A `string` value identifying the unique ID associated with the audit in the Rights Management service.

**EventNamespace:**

A `string` value identifying the namespace qualifier for the event type.

**EventType:**

A `string` value representing unique values within an event namespace.

**EventDate:**

A `dateTime` value representing the time when the event occurred.

**EventDetail:**

A `string` value representing the message string containing the event information.

**PrincipalName:**

A `string` value representing the name of the authenticated principal that was used to cause the event to occur.

**PrincipalUID:**

A `string` value representing the unique ID of the authenticated principal that was used to cause the event to occur.

**WasAllowed:**

A `boolean` value representing whether the action was successful or failed.

**WasOnline:**

A `boolean` value representing whether the operation happened online or offline.

**ClientIPAddress:**

A `string` value representing the IP address of the client that connects to the server.

**LicenseID:**

A `string` value representing the unique ID of the license associated with the secured document in the Rights Management service.

**PolicyID:**

A `string` value representing the unique ID for the policy in the Rights Management service.

**Deactivation:**

Upon notification

**Event Message:**

None

**RMPolicySetEvent**

Rights Management policy set event related to the creation, deletion, and modification of a policy set.

**NOTE:** To use this event type, you need to enable the Rights Management event handlers on the AEM Forms Server. (See Activatingand deactivating event types.)

**Priority:**

3

**Severity:**

Medium

**Event Data:**

Audit event data for a single event.

**EventID:**

A `string` value identifying the unique ID associated with the audit in the Rights Management service.

**EventNamespace:**

A `string` value identifying the namespace qualifier for the event type.

**EventType:**

A `string` value representing unique values within an event namespace.

**EventDate:**

A `dateTime` value representing the time when the event occurred.

**EventDetail:**

A `string` value representing the message string containing the event information.

**PrincipalName:**

A `string` value representing the name of the authenticated principal that was used to cause the event to occur.

**PrincipalUID:**

A `string` value representing the unique ID of the authenticated principal that was used to cause the event to occur.

**WasAllowed:**

A `boolean` value representing whether the action was successful or failed.

**WasOnline:**

A `boolean` value representing whether the operation happened online or offline.

**ClientIPAddress:**

A `string` value representing the IP address of the client that connects to the server.

**LicenseID:**

A `string` value representing the unique ID of the license associated with the secured document in the Rights Management service.

**PolicyID:**

A `string` value representing the unique ID for the policy in the Rights Management service.

**Deactivation:**

Upon notification

**Event Message:**

None

**RMServerEvent**

Rights Management server event related to server configuration changes.

**NOTE:** To use this event type, you need to enable the Rights Management event handlers on the AEM Forms Server. (See Activatingand deactivating event types.)

**Priority:**

3

**Severity:**

Medium

**Event Data:**

Audit event data for a single event.

**EventID:**

A `string` value identifying the unique ID associated with the audit in the Rights Management service.

**EventNamespace:**

A `string` value identifying the namespace qualifier for the event type.

**EventType:**

A `string` value representing unique values within an event namespace.

**EventDate:**

A `dateTime` value representing the time when the event occurred.

**EventDetail:**

A `string` value representing the message string containing the event information.

**PrincipalName:**

A `string` value representing the name of the authenticated principal that was used to cause the event to occur.

**PrincipalUID:**

A `string` value representing the unique ID of the authenticated principal that was used to cause the event to occur.

**WasAllowed:**

A `boolean` value representing whether the action was successful or failed.

**WasOnline:**

A `boolean` value representing whether the operation happened online or offline.

**ClientIPAddress:**

A `string` value representing the IP address of the client that connects to the server.

**LicenseID:**

A `string` value representing the unique ID of the license associated with the secured document in the Rights Management service.

**PolicyID:**

A `string` value representing the unique ID for the policy in the Rights Management service.

**Deactivation:**

Upon notification

**Event Message:**

None

**RMUserEvent**

Rights Management user event related to rights management authentication.

**NOTE:** To use this event type, you need to enable the Rights Management event handlers on the AEM Forms Server. (See Activatingand deactivating event types.)

**Priority:**

3

**Severity:**

Medium

**Event Data:**

Audit event data for a single event.

**EventID:**

A `string` value identifying the unique ID associated with the audit in the Rights Management service.

**EventNamespace:**

A `string` value identifying the namespace qualifier for the event type.

**EventType:**

A `string` value representing unique values within an event namespace.

**EventDate:**

A `dateTime` value representing the time when the event occurred.

**EventDetail:**

A `string` value representing the message string containing the event information.

**PrincipalName:**

A `string` value representing the name of the authenticated principal that was used to cause the event to occur.

**PrincipalUID:**

A `string` value representing the unique ID of the authenticated principal that was used to cause the event to occur.

**WasAllowed:**

A `boolean` value representing whether the action was successful or failed.

**WasOnline:**

A `boolean` value representing whether the operation happened online or offline.

**ClientIPAddress:**

A `string` value representing the IP address of the client that connects to the server.

**LicenseID:**

A `string` value representing the unique ID of the license associated with the secured document in the Rights Management service.

**PolicyID:**

A `string` value representing the unique ID for the policy in the Rights Management service.

**Deactivation:**

Upon notification

**Event Message:**

None

**StartReviewStage**

Content Services review and commenting event thrown explicitly in the process. It starts the next review stage of the Content Services multi-staged shared review.

**Priority:**

1

**Severity:**

Medium

**Event Data:**

Audit event data for a single event.

**ReviewDocumentNodeID:**

A `string` value representing the Content Services node ID of the document to review in the next stage of the shared review.

**InitiatorAuthority:**

A `string` value representing the name of the review initiator. The format of this name is *user-name/lc-domain-name*. For example, `Administrator/DefaultDom`

**Deactivation:**

Upon notification

**Event message:**

None

**TaskAttachmentAdded**

An asynchronous message event indicating that a task attachment has been added.

**Priority:**

3

**Severity:**

Medium

**Event Data:**

**TaskEventType:**

A `short` value representing the node name for the task event type.

**TaskID:**

A `long` value representing the unique ID of the task.

**QueueId:**

A `long` value representing the node name for the associated queue identifier for the task.

**ActionID:**

A `long` value representing the unique ID of the associated operation instance.

**AssignedUser:**

A `string` value representing the user's principal ID (GUID), not the login or common name.

**LongLivedJobId:**

A `string` value representing the node name for the job identifier for the task.

**AttachmentId:**

A `string` value representing the node name for the attachment identifier.

**AttachmentType:**

A `string` value representing the node name for the attachment type. The values can be `attachment` or `note`.

**ProcessInstanceID:**

A `long` value representing the unique process instance ID associated with the task.

**ProcessName:**

A `string` value representing the name of the process this task belongs to.

**StepName:**

A `string` value representing the name of the operation associated with the task. This is the name specified for the AssignTask operation in the process diagram.

**Deactivation:**

Upon notification

**Event Message:**

None

**TaskAttachmentDeleted**

An asynchronous message event indicating that a task attachment has been deleted.

**Priority:**

3

**Severity:**

Medium

**Event Data:**

**TaskEventType:**

A `short` value representing the node name for the task event type.

**TaskID:**

A `long` value representing the unique ID of the task.

**QueueId:**

A `long` value representing the node name for the associated queue identifier for the task.

**ActionID:**

A `long` value representing the unique ID of the associated operation instance.

**AssignedUser:**

A `string` value representing the user's principal ID (GUID), not the login or common name.

**LongLivedJobId:**

A `string` value representing the node name for the job identifier for the task.

**AttachmentId:**

A `string` value representing the node name for the attachment identifier.

**AttachmentType:**

A `string` value representing the node name for the attachment type. The values can be `attachment` or `note`.

**ProcessInstanceID:**

A `long` value representing the unique process instance ID associated with the task.

**ProcessName:**

A `string` value representing the name of the process this task belongs to.

**StepName:**

A `string` value representing the name of the operation associated with the task. This is the name specified for the AssignTask operation in the process diagram.

**Deactivation:**

Upon notification

**Event Message:**

None

**TaskAttachmentInfoUpdated**

An asynchronous message event indicating that a task attachment has had its information updated.

**Priority:**

3

**Severity:**

Medium

**Event Data:**

**TaskEventType:**

A `short` value representing the node name for the task event type.

**TaskID:**

A `long` value representing the unique ID of the task.

**QueueId:**

A `long` value representing the node name for the associated queue identifier for the task.

**ActionID:**

A `long` value representing the unique ID of the associated operation instance.

**AssignedUser:**

A `string` value representing the user's principal ID (GUID), not the login or common name.

**LongLivedJobId:**

A `string` value representing the node name for the job identifier for the task.

**AttachmentId:**

A `string` value representing the node name for the attachment identifier.

**AttachmentType:**

A `string` value representing the node name for the attachment type. The values can be `attachment` or `note`.

**ProcessInstanceID:**

A `long` value representing the unique process instance ID associated with the task.

**ProcessName:**

A `string` value representing the name of the process this task belongs to.

**StepName:**

A `string` value representing the name of the operation associated with the task. This is the name specified for the AssignTask operation in the process diagram.

**Deactivation:**

Upon notification

**Event Message:**

None

### TaskAttachmentUpdated

An asynchronous message event indicating that a task attachment has been updated.

**Priority:**

3

**Severity:**

Medium

**Event Data:**

**TaskEventType:**

A `short` value representing the node name for the task event type.

**TaskID:**

A `long` value representing the unique ID of the task.

**QueueId:**

A `long` value representing the node name for the associated queue identifier for the task.

**ActionID:**

A `long` value representing the unique ID of the associated operation instance.

**AssignedUser:**

A `string` value representing the user's principal ID (GUID), not the login or common name.

**LongLivedJobId:**

A `string` value representing the node name for the job identifier for the task.

**AttachmentId:**

A `string` value representing the node name for the attachment identifier.

**AttachmentType:**

A `string` value representing the node name for the attachment type. The values can be `attachment` or `note`.

**ProcessInstanceID:**

A `long` value representing the unique process instance ID associated with the task.

**ProcessName:**

A `string` value representing the name of the process this task belongs to.

**StepName:**

A `string` value representing the name of the operation associated with the task. This is the name specified for the AssignTask operation in the process diagram.

**Deactivation:**

Upon notification

**Event Message:**

None

**TaskCancelled**

An asynchronous message event indicating that a task has been canceled.

**Priority:**

3

**Severity:**

Medium

**Event Data:**

**TaskEventType:**

A `short` value representing the node name for the task event type.

**TaskID:**

A `long` value representing the unique ID of the task.

**QueueId:**

A `long` value representing the node name for the associated queue identifier for the task.

**ActionID:**

A `long` value representing the unique ID of the associated operation instance.

**AssignedUser:**

A `string` value representing the user's principal ID (GUID), not the login or common name.

**LongLivedJobId:**

A `string` value representing the node name for the job identifier for the task.

**ProcessInstanceID:**

A `long` value representing the unique process instance ID associated with the task.

**ProcessName:**

A `string` value representing the name of the process this task belongs to.

**StepName:**

A `string` value representing the name of the operation associated with the task. This is the name specified for the AssignTask operation in the process diagram.

**Deactivation:**

Upon notification

**Event Message:**

None

**TaskClaimed**

An asynchronous message event indicating that a task has been claimed.

**Priority:**

3

**Severity:**

Medium

**Event Data:**

**TaskEventType:**

A `short` value representing the node name for the task event type.

**TaskID:**

A `long` value representing the unique ID of the task.

**QueueId:**

A `long` value representing the node name for the associated queue identifier for the task.

**ActionID:**

A `long` value representing the unique ID of the associated operation instance.

**AssignedUser:**

A `string` value representing the user's principal ID (GUID), not the login or common name.

**LongLivedJobId:**

A `string` value representing the node name for the job identifier for the task.

**ProcessInstanceID:**

A `long` value representing the unique process instance ID associated with the task.

**ProcessName:**

A `string` value representing the name of the process this task belongs to.

**StepName:**

A `string` value representing the name of the operation associated with the task. This is the name specified for the AssignTask operation in the process diagram.

**Deactivation:**

Upon notification

**Event Message:**

None

**TaskCompleted**

An asynchronous message event indicating that a task has completed.

**Priority:**

3

**Severity:**

Medium

**Event Data:**

**TaskEventType:**

A `short` value representing the node name for the task event type.

**TaskID:**

A `long` value representing the unique ID of the task.

**QueueId:**

A `long` value representing the node name for the associated queue identifier for the task.

**ActionID:**

A `long` value representing the unique ID of the associated operation instance.

**AssignedUser:**

A `string` value representing the user's principal ID (GUID), not the login or common name.

**LongLivedJobId:**

A `string` value representing the node name for the job identifier for the task.

**ProcessInstanceID:**

A `long` value representing the unique process instance ID associated with the task.

**ProcessName:**

A `string` value representing the name of the process this task belongs to.

**StepName:**

A `string` value representing the name of the operation associated with the task. This is the name specified for the AssignTask operation in the process diagram.

**SelectedRoute:**

A `string` value representing the route selected for the task.

**Deactivation:**

Upon notification

**Event Message:**

None

**TaskCompletedWithData**

An asynchronous message event indicating that a task has completed with form data.

**Priority:**

3

**Severity:**

Medium

**Event Data:**

**TaskEventType:**

A `short` value representing the node name for the task event type.

**TaskID:**

A `long` value representing the unique ID of the task.

**QueueId:**

A `long` value representing the node name for the associated queue identifier for the task.

**ActionID:**

A `long` value representing the unique ID of the associated operation instance.

**AssignedUser:**

A `string` value representing the user's principal ID (GUID), not the login or common name.

**LongLivedJobId:**

A `string` value representing the node name for the job identifier for the task.

**ProcessInstanceID:**

A `long` value representing the unique process instance ID associated with the task.

**ProcessName:**

A `string` value representing the name of the process this task belongs to.

**StepName:**

A `string` value representing the name of the operation associated with the task. This is the name specified for the AssignTask operation in the process diagram.

**SelectedRoute:**

A `string` value representing the route selected for the task.

**Deactivation:**

Upon notification

**Event Message:**

None

**TaskConsulted**

An asynchronous message event indicating that a task has been consulted.

**Priority:**

3

**Severity:**

Medium

**Event Data:**

**TaskEventType:**

A `short` value representing the node name for the task event type.

**TaskID:**

A `long` value representing the unique ID of the task.

**QueueId:**

A `long` value representing the node name for the associated queue identifier for the task.

**ActionID:**

A `long` value representing the unique ID of the associated operation instance.

**AssignedUser:**

A `string` value representing the user's principal ID (GUID), not the login or common name.

**LongLivedJobId:**

A `string` value representing the node name for the job identifier for the task.

**ProcessInstanceID:**

A `long` value representing the unique process instance ID associated with the task.

**ProcessName:**

A `string` value representing the name of the process this task belongs to.

**StepName:**

A `string` value representing the name of the operation associated with the task. This is the name specified for the AssignTask operation in the process diagram.

**Deactivation:**

Upon notification

**Event Message:**

None

**TaskCreated**

An asynchronous message event indicating that a task has been created.

**Priority:**

3

**Severity:**

Medium

**Event Data:**

**TaskEventType:**

A `short` value representing the node name for the task event type.

**TaskID:**

A `long` value representing the unique ID of the task.

**QueueId:**

A `long` value representing the node name for the associated queue identifier for the task.

**ActionID:**

A `long` value representing the unique ID of the associated operation instance.

**AssignedUser:**

A `string` value representing the user's principal ID (GUID), not the login or common name. AssignedUser is a value of `null` when assigned to a group.

**AssignedGroup:**

A `string` value representing the group's principal ID (GUID). The AssignedGroup is available only when the task is assigned to a group.

**LongLivedJobId:**

A `string` value representing the node name for the job identifier for the task.

**ProcessInstanceID:**

A `long` value representing the unique process instance ID associated with the task.

**ProcessName:**

A `string` value representing the name of the process this task belongs to.

**StepName:**

A `string` value representing the name of the operation associated with the task. This is the name specified for the AssignTask operation in the process diagram.

**Deactivation:**

Upon notification

**Event Message:**

None

**TaskDeadlined**

An asynchronous message event indicating that a task has been deadlined.

**Priority:**

3

**Severity:**

Medium

**Event Data:**

**TaskEventType:**

A `short` value representing the node name for the task event type.

**TaskID:**

A `long` value representing the unique ID of the task.

**QueueId:**

A `long` value representing the node name for the associated queue identifier for the task.

**ActionID:**

A `long` value representing the unique ID of the associated operation instance.

**AssignedUser:**

A `string` value representing the user's principal ID (GUID), not the login or common name.

**LongLivedJobId:**

A `string` value representing the node name for the job identifier for the task.

**ProcessInstanceID:**

A `long` value representing the unique process instance ID associated with the task.

**ProcessName:**

A `string` value representing the name of the process this task belongs to.

**StepName:**

A `string` value representing the name of the operation associated with the task. This is the name specified for the AssignTask operation in the process diagram.

**Deactivation:**

Upon notification

**Event Message:**

None

**TaskEscalated**

An asynchronous message event indicating that a task has been escalated.

**Priority:**

3

**Severity:**

Medium

**Event Data:**

**TaskEventType:**

A `short` value representing the node name for the task event type.

**TaskID:**

A `long` value representing the unique ID of the task.

**QueueId:**

A `long` value representing the node name for the associated queue identifier for the task.

**ActionID:**

A `long` value representing the unique ID of the associated operation instance.

**AssignedUser:**

A `string` value representing the user's principal ID (GUID), not the login or common name. AssignedUser is a value of `null` when assigned to a group.

**AssignedGroup:**

A `string` value representing the group's principal ID (GUID). The AssignedGroup is available only when the task is assigned to a group.

**LongLivedJobId:**

A `string` value representing the node name for the job identifier for the task.

**ProcessInstanceID:**

A `long` value representing the unique process instance ID associated with the task.

**ProcessName:**

A `string` value representing the name of the process this task belongs to.

**StepName:**

A `string` value representing the name of the operation associated with the task. This is the name specified for the AssignTask operation in the process diagram.

**Deactivation:**

Upon notification

**Event Message:**

None

**TaskFormDataSaved**

An asynchronous message event indicating that the form data for a task has been saved.

**Priority:**

3

**Severity:**

Medium

**Event Data:**

**TaskEventType:**

A `short` value representing the node name for the task event type.

**TaskID:**

A `long` value representing the unique ID of the task.

**QueueId:**

A `long` value representing the node name for the associated queue identifier for the task.

**ActionID:**

A `long` value representing the unique ID of the associated operation instance.

**AssignedUser:**

A `string` value representing the user's principal ID (GUID), not the login or common name.

**LongLivedJobId:**

A `string` value representing the node name for the job identifier for the task.

**ProcessInstanceID:**

A `long` value representing the unique process instance ID associated with the task.

**ProcessName:**

A `string` value representing the name of the process this task belongs to.

**StepName:**

A `string` value representing the name of the operation associated with the task. This is the name specified for the AssignTask operation in the process diagram.

**Deactivation:**

Upon notification

**Event Message:**

None

**TaskForwarded**

An asynchronous message event indicating that a task has been forwarded.

**Priority:**

3

**Severity:**

Medium

**Event Data:**

**TaskEventType:**

A `short` value representing the node name for the task event type.

**TaskID:**

A `long` value representing the unique ID of the task.

**QueueId:**

A `long` value representing the node name for the associated queue identifier for the task.

**ActionID:**

A `long` value representing the unique ID of the associated operation instance.

**AssignedUser:**

A `string` value representing the user's principal ID (GUID), not the login or common name.

**LongLivedJobId:**

A `string` value representing the node name for the job identifier for the task.

**ProcessInstanceID:**

A `long` value representing the unique process instance ID associated with the task.

**ProcessName:**

A `string` value representing the name of the process this task belongs to.

**StepName:**

A `string` value representing the name of the operation associated with the task. This is the name specified for the AssignTask operation in the process diagram.

**Deactivation:**

Upon notification

**Event Message:**

None

**TaskLocked**

An asynchronous message event indicating that a task has been locked.

**Priority:**

3

**Severity:**

Medium

**Event Data:**

**TaskEventType:**

A `short` value representing the node name for the task event type.

**TaskID:**

A `long` value representing the unique ID of the task.

**QueueId:**

A `long` value representing the node name for the associated queue identifier for the task.

**ActionID:**

A `long` value representing the unique ID of the associated operation instance.

**AssignedUser:**

A `string` value representing the user's principal ID (GUID), not the login or common name.

**LongLivedJobId:**

A `string` value representing the node name for the job identifier for the task.

**ProcessInstanceID:**

A `long` value representing the unique process instance ID associated with the task.

**ProcessName:**

A `string` value representing the name of the process this task belongs to.

**StepName:**

A `string` value representing the name of the operation associated with the task. This is the name specified for the AssignTask operation in the process diagram.

**Deactivation:**

Upon notification

**Event Message:**

None

**TaskReassigned**

An asynchronous message event indicating that a task has been reassigned.

**Priority:**

3

**Severity:**

Medium

**Event Data:**

**TaskEventType:**

A `short` value representing the node name for the task event type.

**TaskID:**

A `long` value representing the unique ID of the task.

**QueueId:**

A `long` value representing the node name for the associated queue identifier for the task.

**ActionID:**

A `long` value representing the unique ID of the associated operation instance.

**AssignedUser:**

A `string` value representing the user's principal ID (GUID), not the login or common name.

**LongLivedJobId:**

A `string` value representing the node name for the job identifier for the task.

**ProcessInstanceID:**

A `long` value representing the unique process instance ID associated with the task.

**ProcessName:**

A `string` value representing the name of the process this task belongs to.

**StepName:**

A `string` value representing the name of the operation associated with the task. This is the name specified for the AssignTask operation in the process diagram.

**Deactivation:**

Upon notification

**Event Message:**

None

**TaskRejected**

An asynchronous message event indicating that a task has been rejected.

**Priority:**

3

**Severity:**

Medium

**Event Data:**

**TaskEventType:**

A `short` value representing the node name for the task event type.

**TaskID:**

A `long` value representing the unique ID of the task.

**QueueId:**

A `long` value representing the node name for the associated queue identifier for the task.

**ActionID:**

A `long` value representing the unique ID of the associated operation instance.

**AssignedUser:**

A `string` value representing the user's principal ID (GUID), not the login or common name.

**LongLivedJobId:**

A `string` value representing the node name for the job identifier for the task.

**ProcessInstanceID:**

A `long` value representing the unique process instance ID associated with the task.

**ProcessName:**

A `string` value representing the name of the process this task belongs to.

**StepName:**

A `string` value representing the name of the operation associated with the task. This is the name specified for the AssignTask operation in the process diagram.

**Deactivation:**

Upon notification

**Event Message:**

None

**TaskReminderSent**

An asynchronous message event indicating that a task reminder has been fired.

**Priority:**

3

**Severity:**

Medium

**Event Data:**

**TaskEventType:**

A `short` value representing the node name for the task event type.

**TaskID:**

A `long` value representing the unique ID of the task.

**QueueId:**

A `long` value representing the node name for the associated queue identifier for the task.

**ActionID:**

A `long` value representing the unique ID of the associated operation instance.

**AssignedUser:**

A `string` value representing the user's principal ID (GUID), not the login or common name.

**LongLivedJobId:**

A `string` value representing the node name for the job identifier for the task.

**ProcessInstanceID:**

A `long` value representing the unique process instance ID associated with the task.

**ProcessName:**

A `string` value representing the name of the process this task belongs to.

**StepName:**

A `string` value representing the name of the operation associated with the task. This is the name specified for the AssignTask operation in the process diagram.

**Deactivation:**

Upon notification

**Event Message:**

None

**TaskShared**

An asynchronous message event indicating that a task has been shared.

**Priority:**

3

**Severity:**

Medium

**Event Data:**

**TaskEventType:**

A `short` value representing the node name for the task event type.

**TaskID:**

A `long` value representing the unique ID of the task.

**QueueId:**

A `long` value representing the node name for the associated queue identifier for the task.

**ActionID:**

A `long` value representing the unique ID of the associated operation instance.

**AssignedUser:**

A `string` value representing the user's principal ID (GUID), not the login or common name. AssignedUser is a value of `null` when assigned to a group.

**ShareUser:**

A `string` value representing the user's principal ID (GUID), not the login or common name.

**LongLivedJobId:**

A `string` value representing the node name for the job identifier for the task.

**ProcessInstanceID:**

A `long` value representing the unique process instance ID associated with the task.

**ProcessName:**

A `string` value representing the name of the process this task belongs to.

**StepName:**

A `string` value representing the name of the operation associated with the task. This is the name specified for the AssignTask operation in the process diagram.

**Deactivation:**

Upon notification

**Event Message:**

None

**TaskTerminated**

An asynchronous message event indicating that a task has been terminated.

**Priority:**

3

**Severity:**

Medium

**Event Data:**

**TaskEventType:**

A `short` value representing the node name for the task event type.

**TaskID:**

A `long` value representing the unique ID of the task.

**QueueId:**

A `long` value representing the node name for the associated queue identifier for the task.

**ActionID:**

A `long` value representing the unique ID of the associated operation instance.

**AssignedUser:**

A `string` value representing the user's principal ID (GUID), not the login or common name.

**LongLivedJobId:**

A `string` value representing the node name for the job identifier for the task.

**ProcessInstanceID:**

A `long` value representing the unique process instance ID associated with the task.

**ProcessName:**

A `string` value representing the name of the process this task belongs to.

**StepName:**

A `string` value representing the name of the operation associated with the task. This is the name specified for the AssignTask operation in the process diagram.

**Deactivation:**

Upon notification

**Event Message:**

None

**TaskUnlocked**

An asynchronous message event indicating that a task has been unlocked.

**Priority:**

3

**Severity:**

Medium

**Event Data:**

**TaskEventType:**

A `short` value representing the node name for the task event type.

**TaskID:**

A `long` value representing the unique ID of the task.

**QueueId:**

A `long` value representing the node name for the associated queue identifier for the task.

**ActionID:**

A `long` value representing the unique ID of the associated operation instance.

**AssignedUser:**

A `string` value representing the user's principal ID (GUID), not the login or common name.

**LongLivedJobId:**

A `string` value representing the node name for the job identifier for the task.

**ProcessInstanceID:**

A `long` value representing the unique process instance ID associated with the task.

**ProcessName:**

A `string` value representing the name of the process this task belongs to.

**StepName:**

A `string` value representing the name of the operation associated with the task. This is the name specified for the AssignTask operation in the process diagram.

**Deactivation:**

Upon notification

**Event Message:**

None

**TaskVisibilityChange**

An asynchronous message event indicating that the visibility flag for a task has changed.

**Priority:**

3

**Severity:**

Medium

**Event Data:**

**TaskEventType:**

A `short` value representing the node name for the task event type.

**TaskID:**

A `long` value representing the unique ID of the task.

**QueueId:**

A `long` value representing the node name for the associated queue identifier for the task.

**ActionID:**

A `long` value representing the unique ID of the associated operation instance.

**AssignedUser:**

A `string` value representing the user's principal ID (GUID), not the login or common name.

**LongLivedJobId:**

A `string` value representing the node name for the job identifier for the task.

**ProcessInstanceID:**

A `long` value representing the unique process instance ID associated with the task.

**ProcessName:**

A `string` value representing the name of the process this task belongs to.

**StepName:**

A `string` value representing the name of the operation associated with the task. This is the name specified for the AssignTask operation in the process diagram.

**Deactivation:**

Upon notification

**Event Message:**

None

**TerminateReview**

Content Services review and commenting event thrown explicitly in the process to indicate termination of the review that is waiting for the next stage to start.

**Priority:**

1

**Severity:**

Medium

**Event Data:**

Audit event data for a single event.

**ReviewDocumentNodeID:**

A `string` value representing the Content Services node ID of the document to review in the next stage of the shared review.

**InitiatorAuthority:**

A `string` value representing the name of the review initiator. The format of this name is *user-name/lc-domain-name*. For example, `Administrator/DefaultDom`

**Deactivation:**

Upon notification

**Event message:**

None

## Exception event types

This category of event types includes a default exception event.

**Exception**

Default exception event.

**Priority:**

Not set

**Severity:**

High

**Event Data:**

**FaultName:**

A `string` value representing the name of the fault.

**FaultSource:**

A `string` value representing the source of the fault.

**FaultMessage:**

A `string` value representing the fault message.

**Deactivation:**

Never

RELATED LINKS:

Asynchronous event types

Timer event types

Managing event types

## Timer event types

This category of event types is used by AEM Forms Server to store custom timer events that you create.

Timer events are notifications about time-based situations. To learn how to create custom timer events, see Managingevent types.

**NOTE:** AEM Forms Server does not provide any default timer events.

RELATED LINKS:

Asynchronous event types

Exception event types

# 18. (Deprecated) Guides

# 19. Using (Deprecated) Document Builder

## 19.1. Getting Started with (Deprecated) Document Builder

(Deprecated) Document Builder is a Workbench perspective where you can create and test assembly descriptors (DDX documents) without working directly in XML.

DDX documents describe the required appearance of a resultant document that is assembled from other documents and assets.

When you create a DDX document or open an existing DDX document, Workbench displays the DDX document in the (Deprecated) Document Builder perspective. DDX documents created by Workbench must be contained in an application. You can also use (Deprecated) Document Builder to edit DDX files on your computer.

### Create a DDX document

1) Start Workbench and log on to your AEM Forms Server.
2) Select File > New > Assembly Descriptor.
3) Provide a name for the DDX document.
4) Select an application to store the DDX document in. (See Workingwith Applications.)
5) (Optional) Click Next and select an assembly descriptor template to use as your starting point.
6) Click Finish.

### Open an existing DDX document in an application

1) Start Workbench and log on to your AEM Forms Server.
2) Select Window > Show View > Applications and then double-click the DDX document.

### Open a DDX document on your computer

1) Start Workbench and log on to your AEM Forms Server.
2) Select File > Open and then browse to the DDX document.

### Copy an existing DDX document into (Deprecated) Document Builder

1) Open the dialog or file that contains the existing DDX document and copy the entire DDX expression into your working buffer. The source can be a file or a process input.
2) Use one of the previous instructions to create or open a DDX document.

3) Select Source (upper-right corner). You will see the XML representation of the DDX document.
4) Delete the existing content XML from the canvas.
5) Paste the contents of your working buffer into the canvas.
6) Save the DDX document.

# 19.2. Assembling PDF Documents

(Deprecated) Document Builder helps you create DDX documents that assemble PDF documents. Using (Deprecated) Document Builder, you can create a DDX document by dragging icons onto a canvas. These icons may represent PDF documents, non-PDF documents, form fragments, or XML data (collectively called *assets*). Other icons represent properties applied to the documents, such as watermarks and bookmark titles. Arrange the icons relative to one another to represent the appearance and characteristics of the resultant document. Behind the scenes, (Deprecated) Document Builder creates a DDX document that reflects the icons on the canvas.

## Create a DDX document that builds a simple PDF document

1) Specify the resultant document by selecting New Result > PDF. In the PDF Result panel, leave the default values unchanged.
2) Drag the PDF icon from the Sources panel onto your canvas. (PDF icons from the Sources panel are called *PDF sources*). Position the icon under the PDF result icon added in the previous step. In the PDF Result panel, leave the default values unchanged.
3) Add two more PDF source icons as described in the previous step. Position the icons to be subordinate to the PDF result icon.

   **TIP:** To delete an icon, right-click the icon and select Delete.
4) Save your work and then click Validate to validate the DDX. (See Validatingthe DDX Document.)
5) Click Preview to view an example of the resultant document. (See Previewing theResult from a DDX Document.)
6) (Optional) Apply properties to the result and PDF source icons. (See SetPDF result and source properties.)

Here is the appearance of the canvas that the previous set of steps create. The resulting DDX document produces a PDF document that is assembled from two source PDF documents.

To see the resulting DDX document, click Source (upper-right corner). Here is the DDX source that these steps create.

```
<DDX xmlns="http://ns.adobe.com/DDX/1.0/">
 <PDF result="Untitled 1">
   <PDF source="sourcePDF1"/>
   <PDF source="sourcePDF2"/>
 </PDF>
 <?ddx-source-hint name="sourcePDF1"?>
 <?ddx-source-hint name="sourcePDF2"?>
</DDX>
```

## Create a DDX document that adds a dynamic watermark

Here are the steps for creating a DDX document that creates a PDF document that adds a watermark to a result PDF. The text in the watermark can be provided at runtime.

1) Specify the resultant document by selecting New Result > PDF. In the PDF Result panel, leave the default values unchanged.
2) Drag the PDF icon from the Sources panel onto your canvas. Position the icon under the PDF result icon added in the previous step. In the PDF Result panel, leave the default values unchanged.
3) Drag a Watermark icon from the Page Content panel. Position the icon on top of the PDF result icon.
4) Select the Watermark icon.
5) Select the Content tab in the Watermark panel and copy the following expression into the white field:

```
<StyledText>
 <p>
   <String url="inputmap:///companyStrings/watermarkString">Temporary
string</String>
```

```
  </p>
</StyledText>
```

**TIP:** To delete an icon, right-click the icon and select Delete.

6) Click Apply.
7) Save your work and then click Validate to validate the DDX. (See Validatingthe DDX Document.)
8) Click Preview to view an example of the resultant document. (See Previewing theResult from a DDX Document.)
9) (Optional) Apply properties to the result and PDF source icons. (See SetPDF result and source properties.)

Here is the appearance of the canvas that the previous steps create. The resulting DDX document produces a PDF document that bears a watermark. The styled text for the watermark is provided by the input map URL `inputmap:///companyStrings/watermarkString`. If the URL is omitted, the default value `Temporary string` is used. The input map allows the text in the watermark to be assigned dynamically.

To see the resulting DDX document, click Source (upper-right corner). Here is the DDX source that these steps create.

```
<DDX xmlns="http://ns.adobe.com/DDX/1.0/">
 <PDF result="Untitled 1">
   <PDF source="sourcePDF1"/>
   <Watermark>
     <StyledText>
       <p>
         <String
url="inputmap:///companyStrings/watermarkString">Temporary
string</String>
       </p>
     </StyledText>
   </Watermark>
 </PDF>
 <?ddx-source-hint name="sourcePDF1"?>
</DDX>
```

## Create a DDX document that builds a PDF Portfolio

A PDF Portfolio extends the capability of PDF packages by adding a customizable user interface (navigator), folders, and welcome pages. The interface can enhance the user experience by taking advantage of localized text strings, custom color schemes, and graphic resources. The PDF Portfolio can also include folders for organizing the files in the portfolio.

Here are the steps for creating a DDX document that creates a simple PDF Portfolio. The steps use the default navigator or a navigator that you supply as a named document. The steps modify the navigator color scheme.

1) Specify the resultant document by selecting New Result > PDF. In the PDF Result panel, leave the default values unchanged.
2) Drag the Portfolio icon from the Document Components panel onto your canvas. Position the icon at a level that is subordinate to the PDF result icon.

   *This icon specifies that the result is a PDF Portfolio.*
3) Drag the Navigator icon from the Document Components panel onto your canvas. Position the icon at a level that is subordinate to the Portfolio icon.
4) Click the Navigator icon. In the simple editor for Navigator, change the XML to this expression and then click OK:

   ```
   <Navigator source="mysrc"/>
   ```

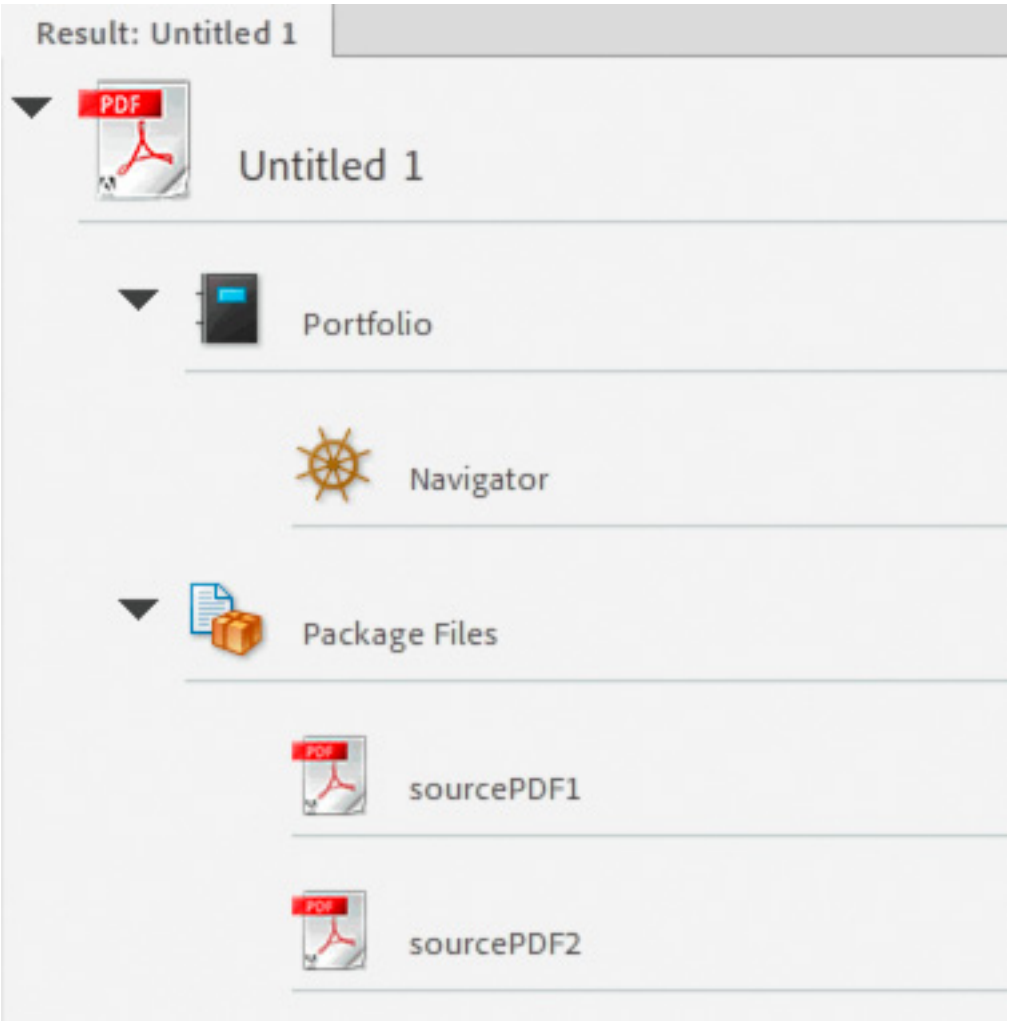
   **TIP:** If you omit the Navigator icon, Acrobat uses a default navigator when it displays the portfolio.
5) To change a characteristic of the portfolio, select the Portfolio icon and use the Simple Editor to change the default value of the element. Here is an example of an XML expression that changes the default color scheme used in the portfolio:

   ```
   <Portfolio>
   <ColorScheme scheme = "pinkScheme"/>
   <Navigator source="mynav"/>
   </Portfolio>
   ```
6) Drag the Package Files icon from the Document Components panel onto your canvas. Position the icon to be subordinate to the PDF result icon. It can be below the Portfolio icon and its Navigator icon. Do not change the values in Simple Editor for Package Files.
7) Drag two or more PDF source icons from the Sources panel onto your canvas. Position the icons to be subordinate to the Package Files icon.

   **TIP:** To achieve the subordination, drag the PDF source icon below and to the right of the Package Files until the ghost image of the PDF source icon appears indented from the Package Files icon.
8) Save your work and then click Validate to validate the DDX file. (See Validating theDDX Document.)
9) Click Preview to view an example of the resultant document. (See Previewing theResult from a DDX Document.)

   **TIP:** You can find several navigators in your Acrobat installation location (for example, C:\Program Files\Adobe\Acrobat 9.0\Acrobat\Navigators).
10) (Optional) Apply properties to the PDF result and PDF source icons. (See Set PDFresult and source properties.)

Here is the appearance of the canvas that the previous set of steps create. The resulting DDX document produces a PDF Portfolio that contains two package files. The PDF Portfolio also contains a description of the PDF Portfolio Layout and contains a navigator (a self-contained user interface).




Here is the DDX source that these steps create.

```
<DDX xmlns="http://ns.adobe.com/DDX/1.0/">
 <PDF result="Untitled 1">
   <Portfolio>
     <Navigator source="mysrc"/>
     <ColorScheme scheme="pinkScheme"/>
   </Portfolio>
   <PackageFiles>
     <PDF source="sourcePDF1"/>
     <PDF source="sourcePDF2"/>
   </PackageFiles>
 </PDF>
 <?ddx-source-hint name="mysrc"?>
 <?ddx-source-hint name="sourcePDF1"?>
 <?ddx-source-hint name="sourcePDF2"?>
</DDX>
```

RELATED LINKS:

Set PDF result and source properties

## Set PDF result and source properties

Use (Deprecated) Document Builder to apply properties to the PDF result and source icons. Examples of these properties are the name of the result, PDF/A profiles, and security. Before you begin these tasks, open a DDX file that produces a PDF result.

1) From the canvas panel, select the PDF result icon.
2) Supply values where needed. (See PDFresult properties you can set from the Result panel.)
3) For each PDF source icon, select the icon and supply values where needed. (See PDFsource properties you can set from the PDF Source panel.)
4) Save your work and then click Validate to validate the DDX file. (See Validating theDDX Document.)
5) Click Preview to view an example of the resultant document. (See Previewing theResult from a DDX Document.)

### PDF result properties you can set from the Result panel

On the Result panel, you can set properties that are applied to the document that is described by the PDF result icon (called the *resultant document*). Select these properties from the Result, FileOptions, and Security tabs.

*Result*

Here are the properties you can set in the Result tab:

**Result:**

Name of the resultant document. The name must be unique among all PDF result icons in the DDX document. This parameter corresponds to the DDX property identified as the `result` attribute of the `PDF` result element.

**TIP:** If the resultant document is returned to the client, other operations in the process can use the specified name to reference the stream.

**Return data to the client:**

When this check box is selected, the resultant document is returned to the client as a stream. Otherwise, the resultant document is available as transient data, which can be referenced as source from within subsequent PDF result icons. This parameter corresponds to the DDX property identified as the `return` attribute of the `PDF` result element.

**Merge document layers:**

When this check box is selected, the layers from different source documents are kept distinct in the assembled document. If you also select the `Layer Label:` property in the PDF source icon, the names are modified by the label; otherwise, the layers keep the same names. This parameter corresponds to the DDX property identified as the `mergeLayers` attribute of the `PDF` result element.

*Layers* are sections of content in a PDF document that document authors or consumers can selectively view or hide. This capability is useful in items such as CAD drawings, layered artwork, maps, and multi-language documents.

**Set Result's Initial view:**

Name of an initial view profile and the properties of the profile. If you have defined the profile in another resultant document, specify the name of that profile but do not specify the initial view properties. This parameter corresponds to the DDX property identified as the `InitialViewProfile` element and the `initialView` attribute of the `PDF` result element.

*File Options*

**Result Document Type:**

This setting determines whether the resultant document is returned as PDF or XDP. Selecting PDF returns the result as a PDF document. Selecting XDP returns the result as an XML Data Packaging (XDP) document. The XDP value can be used only if the base document is an XFA-based PDF document. This parameter corresponds to the DDX property identified as the `format` attribute of the `PDF` result element.

**File Save Option:**

After users fill in fields or otherwise modify the resultant document, they direct Acrobat to save the modified file. This property specifies how Acrobat saves the modifications in the resultant document.

**Incremental:**

Acrobat places changes to the document at the end of the file. The bytes corresponding to the original file are unchanged. Use this setting if you are creating a PDF file that contains one or more signature fields.

**Full:**

Acrobat overwrites any existing incremental saves. This option does not result in optimization for fast web viewing.

**FastWebView:**

Acrobat optimizes and restructures the PDF document for page-at-a-time downloading from web servers. This save causes the removal of any existing incremental saves.

If this attribute is not specified, the save characteristics depend on the PDF level. For PDF 1.4 and later, the DDX processor performs an incremental save, which is relative to the base document. For PDF 1.3 and earlier, the Assembler service performs a full save. This parameter corresponds to the DDX property identified as the `save` attribute of the `PDF` result element.

**TIP:** The removal of existing incremental saves invalidates signatures, certification, and reader-enabled set on the base document. For example, If a user certifies a PDF document that has the Full file save option, then the certification is invalidated when the user saves the document.

**PDF/A Profile Name:**

The name of a PDFAProfile that specifies the type of PDF archive to which the result must conform. A value of `Default` specifies that all the PDFAProfile defaults are used. This parameter corre-

sponds to the DDX property identified as the `PDFAProfile` element and the `pdfa` attribute of the `PDF` result element.

*Security*

**Remove all certifying signatures:**

Removes certifying signatures in the resultant document. If this property is selected and if forms are not removed in the resultant document, the certifying signature can come only from the base source document. Forms are removed by the placing the NoForms or NoXFA property in the resultant document. Those properties are available on the Page Content panel. The Remove All Certifying Signatures property corresponds to the `certification` attribute of the `PDF` result element.

**TIP:** To use a DDX document that removes forms from the resultant document, the Output service must be installed on your AEM Forms Server.

**Remove all reader rights:**

Removes Adobe Reader usage rights, such as the ability to use Adobe Reader to sign, print, or submit a PDF document online. This parameter corresponds to the DDX property identified as the `readerUsageRights` attribute of the `PDF` result element.

**Modify encryption for result:**

Adds or removes encryption from the resultant document. If encryption is added, specify the name of an encryption profile and the properties of the profile. If you have defined the profile in another resultant document within the DDX, specify the name of that profile but do not specify the encryption properties. This parameter corresponds to the DDX property identified as the `PasswordEncryptionProfile` element and the `encryption` attribute of the `PDF` result element.

**PDF source properties you can set from the PDF Source panel**

In the PDF Source panel, you can set properties from the Source, Basic, Tableof Contents, and Miscellaneous tabs. Examples of these properties are the name of the result, PDF/A profiles, and security.

*Source*

**Source:**

Name of a single input data stream, an ordered list of data streams, or an external data URL. Each stream or URL must resolve to a document that is PDF or that your AEM Forms Server installation can convert to PDF. If your installation includes the Generate PDF service, the document can be any of the file types this service can convert (depending on installation and configuration options). This parameter corresponds to the DDX property identified as the `source` attribute of the `PDF` source element.

**TIP:** Instead of specifying this parameter, you can specify the `Doc match reg expression`: *parameter.*

**Password to open file(s):**

Password of the files identified by the PDF source icon. The password is used only if the files are encrypted. If the document is changed before it is added to the resultant document, specify the master password. (See "Working with Secured Documents," in the Assembler Service and DDX Reference.)

**Doc match reg expression:**

Selects source names and their associated data streams from the inputs map. Depending on the `Match Mode:` property, the matched documents are either included in or excluded from the assembled document. If more than one name matches, the names are sorted, as specified in the `Sort Order:` and `Sort Locale:` properties. The Assembler service supports the Java regular expressions. (See the Java tutorial Lesson: Regular Expressions.)

**TIP:** Instead of specifying this parameter, you can specify the `Source:` parameter. If you also specify the `Source:` *parameter, the document is included only if it matches the regular expression regarding the match mode sense.*

**Match Mode:**

Specifies whether the match results are included in or excluded from the document assembly. Select Include to include the matched data streams. Select Exclude to exclude the matched data streams. This parameter applies only when you also provide the `Doc match expression:` parameter.

**Sort Order:**

If the regular expression specified in the `Doc match expression:` parameter matches multiple documents, this attribute specifies the order that those documents are in sorted to create an ordered list of documents. This attribute is not used if the source attribute is specified and it matches an entry in the inputs map. Select Ascending to sort the matched documents in the A-Z ascending order. Select Descending to sort the matched documents in the Z-A descending order. This parameter corresponds to the DDX property identified as the `sortOrder` attribute of the `PDF` source element.

**Sort Locale:**

Locale for sorting the names matched by the `Match Mode:` property. If you provide a value for this property, the value must contain a valid two-character ISO language code (see ISO 639). If you omit this parameter, the sort locale is obtained from the DDX `TargetLocale` element. (Deprecated) Document Builder does not provide an interface for setting the `TargetLocale` element. However, you can use the Source mode to specify this element by directly editing the DDX document. This parameter corresponds to the DDX property identified as the `sortLocale` attribute of the `PDF` source element and the `sortLocale` attribute of the `PDF` source element.

**Include subfolders:**

If true, all files in the folder and subfolders are included. The result is a list of documents for the PDF source that maintains the original folder structure. If false, only the files in the specified folder are included. This parameter is used when the `Doc match expression:` parameter specifies

subfolders. This parameter corresponds to the DDX property identified as the `includeSubFolders` attribute of the `PDF` source element.

**Include all docs:**

Select this field to include files in the folder and subfolders. The result is a list of documents for the PDF source element that maintains the original folder structure. Deselect this field to include only the files in the specified folder. This parameter is used when the `Doc match expression:` parameter specifies folders. This parameter corresponds to the DDX property identified as the `select` attribute of the `PDF` source element.

**Just include these docs:**

Range of documents to include from the ordered list specified in the`Source:` parameter or obtained by applying the `Doc match expression:` property. Leave this field blank if all documents are included. Here are examples of document ranges:

- `1,3,5` Includes the first, third, and fifth documents from the ordered list of documents.
- `1-5,8-10` Includes the first through fifth and eighth through tenth documents from the ordered list of documents.
- `8-last` Includes the eighth through last documents from the ordered list of documents.

This parameter is used when the `Doc match expression:` parameter specifies folders and subfolders. See “Page and document ranges” in the Assembler Service and DDX Reference. This parameter corresponds to the DDX property identified as the `select` attribute of the `PDF` source element.

This parameter corresponds to the DDX property identified as the `PDF` source element’s `select` attribute.

*Basic*

**This is the Base Document:**

Select this check box if the PDF source is the base document. The base document provides the initial structure that the Assembler service uses to set certain document-level properties of the resultant document. These properties include document properties, form data, document-level JavaScript code, and viewer preferences. The resultant document can contain only one source identified as a base document. Documents other than the base document only contribute pages, document components (such as bookmarks, links, file attachments), page labels, page content, and page properties. This parameter corresponds to the DDX property identified as the `baseDocument` attribute of the `PDF` source element.

**There must be at least one valid source:**

Select this check box if the PDF source must include at least one valid PDF document. This parameter corresponds to the DDX property identified as the `required` attribute of the `PDF` source element.

**Page Range:**

Range of pages to include from the source document. Leave this field blank if all documents are included. This parameter corresponds to the DDX property identified as the `pages` attribute of the `PDF` source element. See "Page and document ranges" in the Assembler Service and DDX Reference.

*Table of Contents*

**Include this document's bookmarks in the Table of Contents:**

Select this check box to include the source document's bookmark in the table of contents in the resultant document. This parameter corresponds to the DDX property identified as the `includeInTOC` attribute of the `PDF` source element.

**Create a top-level Bookmark for this source:**

Select this check box to provide a bookmark title.

**Bookmark Title:**

A bookmark title that identifies the source document bookmarks. The string value can be specified with an External Data URL that resolves to a string. This parameter corresponds to the DDX property identified as the `bookmarkTitle` attribute of the `PDF` source element.

*Miscellaneous*

**Layer Label:**

A top-level label under which all the layers of the source document are grouped. This label creates a resultant document that helps users distinguish between layers from different source documents. This label lets you avoid name conflicts in resultant documents. This parameter corresponds to the DDX property identified as the `layerLabel` attribute of the `PDF` source element.

**Link Aliases:**

Alternative names for the parent document for use as link destinations. For each link alias for the PDF source, type the name in the Link Aliases field, and then click Add. This parameter corresponds to the DDX property identified as the `LinkAlias` element.

For more information, see the Assembler Service and DDX Reference.

## 19.3. Assembling XDP Documents

(Deprecated) Document Builder lets you create DDX documents that assemble XDP documents. An XML Data Package (XDP) is a container for multiple related XML documents. Here are the XML documents that can be included in an XDP document:

- XML Forms Architecture (XFA) form, which is also called a form design
- Custom data to merge with the form
- Other XML data used to process the form

Designer saves form designs as XDP files.

## Create a DDX document that builds a simple XDP document

Here are the general steps for creating a DDX document that creates an XDP document that is made from other XDP documents.

1) Specify the resultant document by selecting New Result > XDP. In the XDP Result panel, leave the default values unchanged.
2) Drag the XDP icon from the Sources panel onto your canvas. Position the icon under the XDP result icon added in the previous step. In the XDP Result panel, leave the default values unchanged.
3) Add two more Source XDP icons as described in the previous step.
4) Save your work and then click Validate to validate the DDX file. (See Validating theDDX Document.)
5) Click Preview to view an example of the resultant document. (See Previewing theResult from a DDX Document.)
6) (Optional) Apply properties to the result and source XDP documents. (See Set PDFresult and source properties.)

Here is the appearance of the canvas that these steps create. The resulting DDX document produces an XDP document that is assembled from three source XDP documents.

Here is the DDX source that these steps create.

```
<DDX xmlns="http://ns.adobe.com/DDX/1.0/">
 <PDF result="Untitled 4"/>
 <XDP result="Untitled 1">
```

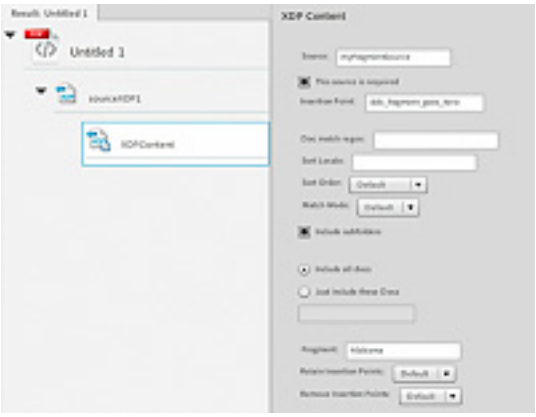
```
    <XDP source="sourceXDP1"/>
    <XDP source="sourceXDP2"/>
    <XDP source="sourceXDP3"/>
  </XDP>
  <?ddx-source-hint name="sourceXDP1"?>
  <?ddx-source-hint name="sourceXDP2"?>
  <?ddx-source-hint name="sourceXDP3"?>
</DDX>
```

## Create a DDX document that inserts a form fragment into an XDP document

Here are the general steps for creating a DDX document that includes a form fragment in an XDP document.

1) Specify the resultant document by selecting New Result > XDP. In the XDP Result panel, leave the default values unchanged.
2) Drag the XDP icon from the Sources panel onto your canvas. Position the icon under the XDP result icon added in the previous step.
3) In both the XDP Result panel and the XDP Source panel, leave the default values unchanged.
4) Drag the XDP Content icon from the Document Components panel onto your canvas. Position the icon to be subordinate to the source XDP icon.
5) With the XDP Content icon selected, supply these values in the XDP Content panel:
   – **Source:** Specify a string or URL that provides the XDP content.
   – **Fragment:** Specify the name of the subform to use from the source XDPContent file. The named subform is the fragment inserted into the parent XDP source document.
   – **Insertion Point:** Specify the name of the insertion point in the XDP source file where the fragment is inserted.
   – **Other fields:** Leave the other fields with the default settings.
6) Save your work and then click Validate to validate the DDX file. (See Validating theDDX Document.)
7) Click Preview to view an example of the resultant document. (See Previewing theResult from a DDX Document.)
8) (Optional) Set XDP result properties. (See SetPDF result and source properties.)
9) (Optional) Set other XDP Content properties. (See SetXDP Content source properties.)

Here is the appearance of the canvas that these steps create. The resulting DDX document produces an XDP document that is assembled from one source XDP document. The fragment in the XDP Content is inserted into the XDP source document. The XDP Content panel shows the properties for the Insertion Point and Fragment. These values mean that the fragment named `Alabama` from the `myFragmentSource` document is inserted into the `ddx_fragment_goes_here` insertion point in the `sourceXDP1` document.




Here is the DDX source that these steps create. The name of the insertion point is `ddx_fragment_goes_here`. The name of the fragment is `Alabama`.

```
<DDX xmlns="http://ns.adobe.com/DDX/1.0/">
 <XDP result="Untitled 1">
   <XDP source="sourceXDP1" fragment="">
     <XDPContent fragment="Alabama"
insertionPoint="ddx_fragment_goes_here" source="myFragmentSource"/>
   </XDP>
 </XDP>
 <?ddx-source-hint name="sourceXDP1"?>
 <?ddx-source-hint name="myFragmentSource"?>
</DDX>
```

## Set XDP result and source properties

Use (Deprecated) Document Builder to apply properties to the XDP result and source documents. For example, these properties select the XDP documents to use and the order in which they are considered. Before you begin these instructions, open a DDX file that produces an XDP result.

1) From the canvas panel, select the result XDP icon.
2) In the XDP Result panel, supply values where needed. (See XDPresult properties you can set from the Result tab.)
3) Save your work and then click Validate to validate the DDX file. (See Validating theDDX Document.)
4) Click Preview to view an example of the resultant document. (See Previewing theResult from a DDX Document.)
5) From the canvas panel, select an XDP source icon.
6) From the XDP Source panel, supply values where needed. (See XDPsource properties you can set from the XDP Source panel.)
7) Save your work and then click Validate to validate the DDX file.
8) Click Preview to view an example of the resultant document.

### XDP result properties you can set from the Result tab

**Result:**

Name of the resultant document. The name must be unique among all resultant documents in the DDX document.

**TIP:** If the XDP result is returned to the client, other operations in the process can use the specified name to reference the stream.

**Return data to the client:**

When this check box is selected, the result XDP document is returned to the client as a stream. Otherwise, the result XDP document is available as transient data, which can be referenced as source from within a subsequent XDP source icon.

**Aggregate XDP Content:**

Select All to add at the insertion point every level of data from the XDP content. (Order of insertion is inner-to-outer level.) Select None to add at the insertion point only the most inner level of data from the XDP Content. Use this property only if your source XDP documents contain forms designed to work with form fragments.

**Retain Insertion Points:**

Specify the insertion points to retain. Select All to retain all of the insertion points. Select None to remove all of the insertion points. To retain specifically named insertion points, select List and then provide the names of each insertion point to retain. Use this property only if your source XDP documents contain forms designed to work with form fragments.

**Remove Insertion Points:**

Specify the insertion points to remove. Select All to remove all of the insertion points. Select None to retain all of the insertion points. To remove specifically named insertion points, select List and then provide the names of each inserting point to retain. Use this property only if your source XDP documents contain forms designed to work with form fragments.

### XDP source properties you can set from the XDP Source panel

**Source:**

Name of a single input data stream, an ordered list of data streams, or an external data URL. Each stream or URL must resolve to a document. If the document is not XDP, the Assembler service tries to convert the document to XDP.

**This source is required:**

Select this check box to terminate XDP assembly if this source does not specify a valid XDP document.

**This is the Base Document:**

Select this check box if the XDP source is the base document. The base document provides the initial structure that the Assembler service uses to set certain document-level properties of the result XDP document. These document-level properties include form data, document-level JavaScript code, and viewer preferences. The resultant document can contain only one source identified as a base document. Documents other than the base document contribute pages, document components (such as bookmarks, links, file attachments), page labels, page content, and page properties to the resultant document.

**Doc match regex:**

Selects source names and their associated data streams from the inputs map. Depending on the `Match Mode:` property, the matched documents are either included or excluded in the assembled document. If more than one name matches, the names are sorted, as specified in the `Sort Locale:` and `Sort Order:` properties. The Assembler service supports the Java regular expressions. (See the Java tutorial Lesson: Regular Expressions.)

**TIP:** Instead of specifying this parameter, you can specify the `Source:` parameter. If you also specify the `Source:` *parameter, the document is included only if it matches the regular expression regarding the match mode sense.*

**Sort Locale:**

Locale to use for sorting the names matched by the `Match Mode:` property. The value of this attribute must be a valid two-character ISO language code (see ISO 639). If you omit this parameter, the Assembler service determines the locale from the DDX `TargetLocale` element. (Deprecated) Document Builder does not provide an interface for setting the `TargetLocale` element. However, you can use the Source mode to specify this element by directly editing the DDX document. (See Editingthe XML for the DDX Document.)

**Sort Order:**

If the regular expression specified in the `Doc match reg expression:` parameter matches multiple documents, this attribute specifies the order in which those documents are sorted. This attribute is not used if the source attribute is specified and it matches an entry in the inputs map. Select Ascending to sort the matched documents in ascending order: A-Z. Select Descending to sort the matched documents in descending order: Z-A.

**Match Mode:**

Specifies whether the match results are included or excluded from the document assembly. Select Include to include the matched data streams. Select Exclude to exclude the matched data streams. This parameter applies only when you also provide the `Doc match reg expression:` parameter.

**Include subfolders:**

If true, all files in the folder and subfolders are included. This results in a list of documents for the PDF source element that maintains the original folder structure. If false, only the files in the speci-

fied folder are included. This parameter is used when the `Doc match reg expression:` parameter specifies subfolders.

**Include all docs:**

Select this field to include files in the folder and subfolders. This results in a list of documents for the PDF source element that maintains the original folder structure. Deselect this field to include only the files in the specified folder. This parameter is used when the `Doc match reg expression:` parameter specifies folders.

**Just include these docs:**

Range of documents to include from the ordered list specified in the `Source:` parameter or obtained by applying the `Doc match reg expression:` property. Leave this field blank if all documents are included. Here are examples of document ranges:

- `1,3,5` Includes the first, third, and fifth documents from the ordered list of documents.
- `1-5,8-10` Includes the first through fifth and eighth through tenth documents from the ordered list of documents.
- `8-last` Includes the eighth through last documents from the ordered list of documents.

This parameter is used when the `Doc match reg expression:` parameter specifies folders and subfolders. See “Page and document ranges” in the Assembler Service and DDX Reference.

This parameter corresponds to the DDX property identified as the `XDP` source element’s `select` attribute.

**Fragment:**

The name of the subform in the XDP source to include in the XDP result. Omit this property if the entire XDP source document is used in the XDP result. This property is invalid when it appears in base XDP sources.

**Retain Insertion Points:**

Specify the insertion points to retain. Select All to retain all of the insertion points. Select None to remove all of the insertion points. To retain specific named insertion points, select List and then provide the names of each insertion point to retain. Use this property only if your source XDP documents contain forms designed to work with form fragments.

**Remove Insertion Points:**

Specify the insertion points to remove. Select All to remove all of the insertion points. Select None to retain all of the insertion points. To remove specific named insertion points, select List and then provide the names of each inserting point to retain. Use this property only if your source XDP documents contain forms designed to work with form fragments.

For more information, see Assembler Service and DDX Reference.

## Set XDP Content source properties

Use (Deprecated) Document Builder to apply properties to XDP Content source files. For example, these properties select the XDP documents to use and the order in which they are considered. Before you begin these instructions, open a DDX file that uses an XDP Content icon.

1) From the canvas panel, select an XDP Content icon.
2) From the XDP Content panel, supply values where needed. (See XDPContent attributes you can set from the XDP Content panel.)
3) Save your work and click Validate to validate the DDX file. (See Validatingthe DDX Document.)
4) Click Preview to view an example of the resultant document. (See Previewing theResult from a DDX Document.)

### XDP Content attributes you can set from the XDP Content panel

**Source:**

Name of a single input data stream, an ordered list of data streams, or an external data URL. Each stream or URL must resolve to a document. If the document is not XDP, the Assembler service tries to convert the document to XDP.

**This source is required:**

Select this check box to terminate XDP assembly if this source does not specify a valid XDP document.

**Insertion Point:**

Name of the insertion point in the XDP source file where the fragment is inserted.

**Doc match regex:**

Selects source names and their associated data streams from the inputs map. Depending on the `Match Mode:` property, the matched documents are either included in or excluded from the assembled document. If more than one name matches, the names are sorted, as specified in the `Sort Locale:` and `Sort Order:` properties. The Assembler service supports the Java regular expressions. (See the Java tutorial Lesson: Regular Expressions.)

**TIP:** Instead of specifying this parameter, you can specify the `Source:` parameter. If you also specify the `Source:` *parameter, the document is included only if it matches the regular expression regarding the match mode sense.*

**Sort Locale:**

Locale to use for sorting the names matched by the `Match Mode:` property. The value of this attribute must be a valid two-character ISO language code (see ISO 639). If you omit this parameter, the Assembler service determines the locale from the DDX `TargetLocale` element. (Deprecated) Document Builder does not provide an interface for setting the `TargetLocale` element. However, you can use the Source mode to specify this element by directly editing the DDX document.

**Sort Order:**

If the regular expression specified in the `Doc match reg expression:` parameter matches multiple documents, this attribute specifies the order in which those documents are sorted. This attribute is not used if the source attribute is specified and it matches an entry in the inputs map. Select Ascending to sort the matched documents in the A-Z ascending order. Select Descending to sort the matched documents in the Z-A descending order.

**Match Mode:**

Specifies whether the match results are included in or excluded from the document assembly. Select Include to include the matched data streams. Select Exclude to exclude the matched data streams. This parameter applies only when you also provide the `Doc match reg expression:` parameter.

**Include subfolders:**

If true, all files in the folder and subfolders are included. The result is a list of documents for the PDF source element that maintains the original folder structure. If false, only the files in the specified folder are included. This parameter is used when the `Doc match reg expression:` parameter specifies subfolders.

**Include all docs:**

Select this field to include files in the folder and subfolders. The result is a list of documents for the PDF source element that maintains the original folder structure. Deselect this field to include only the files in the specified folder. This parameter is used when the `Doc match reg expression:` parameter specifies folders.

**Just include these docs:**

Range of documents to include from the ordered list specified in the `Source:` parameter or obtained by applying the `Doc match reg expression:` property. Leave this field blank if all documents are included. Here are examples of document ranges:

- `1,3,5` Includes the first, third, and fifth documents from the ordered list of documents.
- `1-5,8-10` Includes the first through fifth and eighth through tenth documents from the ordered list of documents.
- `8-last` Includes the eighth through last documents from the ordered list of documents.

This parameter is used when the `Doc match reg expression:` parameter specifies folders and subfolders. This parameter corresponds to the DDX property identified as the `XDP` source element's `select` attribute. See "Page and document ranges" in the Assembler Service and DDX Reference.

**Fragment:**

A string that identifies the fragment to name to insert. To insert the entire form at the insertion point, leave this field blank. Use this property only if your source XDP documents contain forms designed to work with form fragments.

**Retain Insertion Points:**

Specify the insertion points to retain. Select All to retain all of the insertion points. Select None to remove all of the insertion points. To retain specific named insertion points, select List and then provide the names of each insertion point to retain. Use this property only if your source XDP documents contain forms designed to work with form fragments.

**Remove Insertion Points:**

Specify the insertion points to remove. Select All to remove all of the insertion points. Select None to retain all of the insertion points. To remove specific named insertion points, select List and then provide the names of each insertion point to retain. Use this property only if your source XDP documents contain forms designed to work with form fragments.

For more information, see Assembler Service and DDX Reference.

## Resolve image references in source

XDP documents can contain images linked either through absolute or relative references. Assembler service, by default, retains the references to the images in the resultant XDP document.

You can specify how the Assembler service handles the images referenced in the source XDP documents when assembling. You can choose to have all the images embedded in the resultant PDF so that it contains no relative or absolute references. You define this by setting the value of the `resolveAssets` tag, which can take any of the following options:

| Value | Description |
|---|---|
| none | Does not resolve any of references. All references are retained. |
| all | Embeds all the images referenced through absolute or relative references in the source XDP document. |
| relative | Embeds all the images referenced through relative references in the source XDP document. |
| absolute | Embeds all the images referenced through absolute references in the source XDP document. |

You can specify the value of the `resolveAssets` attribute either in the XDP source tag or in the parent XDP result tag. The attribute specified for the XDP result tag is inherited by all the XDP source elements which are children of the XDP result. However, explicitly specifying the attribute for a source element overrides the setting of the result element for that source document alone.

### Resolve all image references in a source XDP document

To convert all image references in the source XDP documents, specify the `resolveAssets` attribute for the result document to `all`, as in the example below:

```
<DDX xmlns="http://ns.adobe.com/DDX/1.0/">
<XDP result="result.xdp" resolveAssets="all">
```

```
<XDP source="input1.xdp" />
<XDP source="input2.xdp" />
<XDP source="input3.xdp" />
</XDP>
</DDX>
```

You can also specify the attribute for all the source XDP documents independently to get the same result.

```
<DDX xmlns="http://ns.adobe.com/DDX/1.0/">
<XDP result="result.xdp">
<XDP source="input1.xdp" resolveAssets="all"/>
<XDP source="input2.xdp" resolveAssets="all"/>
<XDP source="input3.xdp" resolveAssets="all"/>
</XDP>
</DDX>
```

**Resolve specified image references in a source XDP document**

You can selectively specify the image references that you want to resolve by specifying the `resolveAssets` attribute for them. The attributes for individual source documents override the resultant XDP document's setting. In this example, the fragments included are also resolved.

```
<DDX xmlns="http://ns.adobe.com/DDX/1.0/">
<XDP result="result.xdp">
<XDP source="input1.xdp" resolveAssets="all">
<XDPContent source="fragment.xdp" insertionPoint="MyInsertionPoint"
fragment="myFragment"/>
</XDP>
<XDP source="input2.xdp" />
</XDP>
</DDX>
```

**Selectively resolve absolute or relative references**

You can selectively resolve absolute or relative references in all or some of the source documents, as shown in the example below:

```
<DDX xmlns="http://ns.adobe.com/DDX/1.0/">
<XDP result="result.xdp" resolveAssets="absolute">
<XDP source="input1.xdp" resolveAssets="relative"/>
<XDP source="input2.xdp" resolveAssets="all"/>
<XDP source="input3.xdp" resolveAssets="absolute" />
</XDP>
</DDX>
```

## 19.4. Validating the DDX Document

(Deprecated) Document Builder can validate your DDX document to ensure that it is consistent with the DDX schema. The schema defines the rules for DDX documents, such as the relationships between DDX elements and the properties those elements can have.

(Deprecated) Document Builder itself cannot prevent you from creating erroneous DDX documents. Validating and previewing your DDX document at various stages is the best way to ensure that your final DDX document produces the results you want.

### Validate the DDX

1) Ensure that the DDX to validate is open in Document Builder.
2) Click Validate.

   *If validation is successful, Document Builder displays a green check mark near the Validate button. If the validation is unsuccessful, (Deprecated) Document Builder displays a pane that lists the errors and their severity.*

### Troubleshooting tips

- Save your work before you validate it.
- For errors with source icons, ensure that you specified a Source property or a Match Source property. You can specify both properties for one icon.

## 19.5. Previewing the Result from a DDX Document

Use (Deprecated) Document Builder to preview the resultant document that AEM Forms produce when the document template is interpreted. Before you can preview a document template, you must provide actual instances of the source documents and assets used in the document template. If all source documents and assets have actual instantiations, you can view the resultant document.

### Set up your system for previewing XDP results

The Preview feature uses your browser to display the result from interpreting the DDX document. By default, browsers use Adobe Acrobat or Adobe Reader to open PDF and XDP file types. However, those programs cannot open XDP files. To view an XDP result, configure your system to use Designer to open XDP files.

### On Windows systems

1) Open a folder.
2) Select Tools> File Options > File Types.
3) Select the XDP extension from the Registered File Types panel and click Change.
4) If Designer is present in the Programs list, select it. Otherwise, click Browse and navigate to the Designer executable installed on your computer.

### Preview the DDX result

1) In (Deprecated) Document Builder, open the DDX document to preview.
2) Ensure the DDX document is valid. (See Validatingthe DDX Document.)
3) Click Preview Result.
4) Provide each file's location in the Select Source Files window. (Deprecated) Document Builder lets you specify the file location by using several conventions:
   – Single file on your computer: Select File and click Browse Identify the location of the file.
   – Folder on your computer: Select Folder and click Browse. Identify the location of the folder.
   – String: Select String and enter the desired value for the string parameter.

   **TIP:** When you use the Folder option to supply values to a Source Match property, the files in the folder and all the files in the subfolders are uploaded to the Assembler service.

   **TIP:** To ensure a reasonable preview of the DDX result, the files you supply must contain the properties referenced from the DDX.
5) To delete entries in the Select Source Files dialog box, select the item (a check mark appears) and then click Delete.
6) Click OK.
7) Click Open in Browser to view the resultant document in your browser.
8) Click Save to save the resultant document or the job log.

Here is an example that shows how to provide an input map URL in the Preview dialog window. For the first source (`sourcePDF1`), select the type, click the Browse button, and select the file. For the second source (`companyStrings/watermarkString`), select the String type and enter the string in the field. You can use these values to preview the DDX document described in Createa DDX document that adds a dynamic watermark. The resultant document bears a watermark with the styled text that the input map URL provides.

## 19.6. Editing the XML for the DDX Document

Use (Deprecated) Document Builder to view and directly edit document templates. (Deprecated) Document Builder has built-in assistants that provide context-sensitive hints on suitable elements or on available attributes for the current element.

### Edit the XML

1) Ensure that the DDX document to edit is open in (Deprecated) Document Builder.
2) Click Source (upper-right corner).
3) To add new content, position the cursor within the XML content and type the content. This behavior is similar to a line editor.
4) To modify existing content, select the content to change and retype the revised text. This behavior is also similar to a line editor.

5) Validate the DDX file by clicking Validate. (See Validatingthe DDX Document.)
6) Preview the result by clicking Preview. (See Previewingthe Result from a DDX Document.)

You cannot type tabs in (Deprecated) Document Builder. As a result, new lines begin at column 1. When (Deprecated) Document Builder redisplays the XML for a modified DDX document, the XML is pretty-printed. (Deprecated) Document Builder redisplays the XML in these situations:

- You close and then open the DDX document.
- You switch from Design and then back to Source.

# 20. Creating device profiles using XDC Editor

## 20.1. About XDC files

XML Forms Architecture XFA Device Control (XDC) Language, commonly called a *device profile* is a printer description file in XML format that makes it possible to output documents as PostScript®, PCL, ZPL, IPL, DPL, and TPCL formats. XDC files are based on an XML-based schema that describe printer capabilities and expose some print options for use by printer device drivers.

XDC files describe printer capabilities such as the printer features and the paper types. Designer creates Adobe XML Form (*.xdp) files. AEM forms uses XDP files to generate the Page Description Language (PDL) that is sent to a network printer.

XDC files describe printer capabilities such as the printer features and the paper types. output and Designer use Adobe Acrobat® XML Data Package (XDP) files. Output uses XDP files to generate the Page Description Language (PDL) that is sent to a network printer. Designer uses the information in XDC files to identify page types that can be selected for master pages.

In Workbench, assets such as XDC files reside inside containers called *applications*. Applications contain all the resources that are required for implementing a AEM forms solution. Storing XDC files inside applications helps you make them available to other developers on an as-needed basis. By default, the Applications view in Workbench shows the applications and assets that are available in your local folder, which were either created by you or copied from the server. You can also view the applications and assets that were checked into the server either by you or by other users. See Workbench Help

for more information about AEM forms applications.

When you want to edit an XDC file, open the XDC file from an application that contains the XDC file or from your local file system. After editing, you can save the updated XDC file in the local file system. You check the updated file back into the server to make it available for AEM forms.

When you want to edit an XDC file, open the XDC file from an application that contains the XDC file or from your local file system. After editing, you can save the updated XDC file in the local file system. You check the updated file back into the server to make it available for Output.

SAP® Interactive Forms by Adobe supports the following PDLs:

AEM forms supports the following PDLs:

- PostScript
- Printer Control Language (PCL)
- Zebra® Printer Language (ZPL)
- Intermec® Printer Language (IPL) for Intermec printers
- Datamax® Printer Language (DPL) for Datamax label printers

- TEC Printer Command Language (TPCL) for Toshiba® TEC label printers

## Role of XDC files

Designer and AEM forms use XDC files. Designer uses a single XDC file to determine the media sizes (paper sizes) and fonts that are available on the target printer. Adobe document services uses the XDC file that is associated with a printer to determine a wider set of printer characteristics.

Designer and Output use XDC files. Designer uses a single XDC file to determine the media sizes (paper sizes) and fonts that are available on the target printer. Output uses the XDC file that is associated with a printer to determine a wider set of printer characteristics.

## XDC file contents

The following list describes the XDC contents that are accessible from XDC Editor. These categories correspond to tabs in the XDC Editor panel:

**Trays**

(PCL and PostScript printers only) You can add or modify input trays in this panel. In addition, you can easily map input trays to different media. You can specify the input and output trays that are available on the printer that is independent of the media types. You can then map input trays to different media types.

**Fonts**

A definition of the fonts that are residing on the printer. Administrators add and remove fonts to match those fonts that are installed on a target printer. When an administrator adds fonts, XDC Editor adds the font metrics and other characteristics to the XDC file.

**Printer capabilities**

Printer features, such as duplex printing and PDL language level. Most of these options depend on the type of PDL indicated by the XDC file.

**Sequences**

PDL sequences are short PDL sequences that Adobe document services injects into the PDL syntax that is sent to the printer at specific points in the print job such as at the beginning of a page or a record.

**Sequences**

PDL sequences are short PDL sequences that Output injects into the PDL syntax that is sent to the printer at specific points in the print job such as at the beginning of a page or a record.

**Mediums**

Media types (paper types) that are available on the printer. Typically, form authors add the mediums to determine the type of paper that the forms must be printed on.

Using Designer, a form author specifies the media type that each page in the form is printed on. When a form is printed, it uses the XDC file for the printer to associate the media type for a page with a specific input tray. It also uses the XDC file to determine the printer-specific commands to select the input and output trays. Adobe document services incorporates these commands into the setup instructions it sends to the printer.

Using Designer, a form author specifies the media type that each page in the form is printed on. When a form is printed, it uses the XDC file for the printer to associate the media type for a page with a specific input tray. It also uses the XDC file to determine the printer-specific commands to select the input and output trays. Output incorporates these commands into the setup instructions it sends to the printer.

Designer and Adobe document services use font metrics for these purposes:

Designer and Output use font metrics for these purposes:

- Designer uses font metric data to find substitute fonts so that it can display forms for which it lacks fonts specified in the form. A font may be unavailable for many reasons. For example, some fonts such as magnetic ink character recognition (MICR) fonts are installed only on the printer. Some fonts may be too expensive to be installed on every computer hosting Designer.
- Adobe document services uses the font metric data to determine the placement of characters on the page.
- Output uses the font metric data to determine the placement of characters on the page.

## Using XDC files

Adobe document services uses XDC files to generate the PDL that is sent to a printer. Adobe document services prints a form by producing PDL instructions that it sends to a printer. The PDL instructions reflect information from a form design file, XML form data, and an XDC file, as shown in the illustration below.

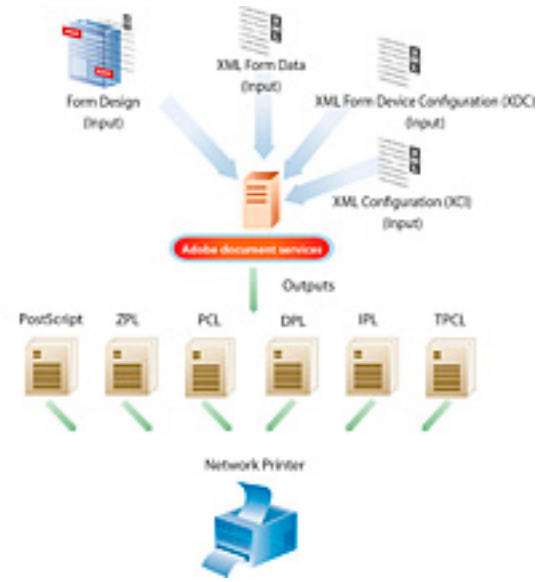
Output uses XDC files to generate the PDL that is sent to a printer. Output prints a form by producing PDL instructions that it sends to a printer. The PDL instructions reflect information from a form design file, XML form data, and an XDC file, as shown in the illustration below.

These files contribute in the following ways:

- The form design file and the XML form data files are merged to produce a static, printable form with data.
- The XDC file provides information about the printer to produce PDL instructions specific for the target printer. For example, the XDC file provides mappings between paper types (media) and input tray numbers. In addition, the XDC file provides special PDL sequences that are injected into the PDL instructions sent to the printer.

For more information about the Output service, see Services Reference

.




## Using XDC files in Designer

Designer uses a single XDC file (Designer.xdc) to determine the allowable selections for paper types (media) and fonts. This file is a superset of the paper sizes and fonts specified by the XDC files that are installed by default. In some situations, update the settings in the Designer.xdc file to be consistent with the settings in the XDC files that your printers use.

Designer also uses printer-specific XDC files to print test copies of forms.

For more information, see DeployingXDC files.

For more information, see DeployingXDC files to Designer installations.

### Selecting a paper type for a master page

From Designer, you can select from media types (paper types) from the Paper Type field. Designer obtains the values for this field from the Designer.xdc file.

1) In Designer, open a form (XDP file).
2) Select Window > Hierarchy to show the Hierarchy panel.
3) Select Palettes > Hierarchy to show the Hierarchy panel.
4) Select Windows > Object to display the Object panel.
5) Select Palettes > Object to display the Object panel.
6) In the Object panel, click the Master Page tab.
7) From the Paper Type list, select the media types that you require. Designer takes each of the values in the Paper Type field from the Designer.xdc file.

### Selecting fonts applied to text

From Designer you can select from the fonts in the Font menu. Designer obtains the values for this menu from the Designer.xdc file.

1) In Designer, open a form (XDP file).
2) Select text in the form.
3) Select Palettes > Font to show the Font palette.
4) Select Windows > Font to show the Font palette.
5) From the Font list, select the font names, sizes, and styles that you require. This list is populated by the list of fonts specified in the Designer.xdc file.

### Determining whether you need to modify the Designer.xdc file

The XDC file that Designer uses (Designer.xdc) defines a set of paper types that can be used for most cases. If you have any of the situations occur, modify the Designer.xdc file to add additional paper types. You also modify the XDC file for the target printer to map the additional paper types to input trays.

- Uses more than four types of any one of the standard paper sizes: letter, A4, or B4 JIS
- Uses more than two types of any one of the other paper sizes defined in the Designer.xdc file
- Uses paper sizes that are not yet defined in the Designer.xdc file

Otherwise, the only requirement is that the correct paper types be installed in the input trays that are associated with the paper type, as defined in the XDC file for the printer.

If you choose to customize paper names that form authors can select, modify the Designer.xdc file and the XDC file for the target printers. Such changes may be preferable to ensure that consistent naming conventions are used within an organization.

See Considerationsfor printing a form using the installed XDC files and Adding newpaper types to the Designer.xdc file.

### Considerations for printing a form using the installed XDC files

Using the installed XDC files, printers can select the paper trays from which they pull basic paper types, such as letter and legal.

#### Single paper type for a paper size

If the printer has one paper tray with plain letter-sized paper and the form’s master pages use letter-sized paper, the printer automatically prints on the letter-sized paper. This practice is also true for legal and other sizes that are installed on the printer.

#### Four or fewer paper types for a paper size

The Designer.xdc file defines a set of special paper types that can be used for most cases before it is necessary to add new paper types. For letter size, these types are the defined paper types:

- Letter Plain
- Letter Letterhead
- Letter Color

- Letter Special

  An equivalent set of paper types is also available for the A4 and B4 JIS paper sizes.

  For example, an application can use letter-sized paper from four or fewer input trays (for example, one for plain paper, one for preprinted letterhead, and one for pink paper) without modifying the Designer.xdc file. The assumption is that the correct paper types are placed in the input trays that are associated with each paper type, as defined in the XDC file for the printer.

  Another example is of a complex job that you can print using the XDC files provided with your installation, without modification. Consider an insurance document package that uses four different letter-sized paper types:

- First page uses letterhead
- Second page through third-to-last pages use plain white
- Second-to-last page uses blue paper with some kind of notice/information
- Final page uses preprinted and prescored insurance cards at the bottom

  As before, the assumption is that the correct paper types are installed in the input trays that are associated with each paper type, as defined in the XDC file for the printer.

### Adding new paper types to the Designer.xdc file

To add new paper types to the Designer.xdc file, use a line editor or XML-capable editor to directly modify the contents of the file. That is, you cannot use XDC Editor for this task.

**NOTE:** Adding new paper types to the Designer.xdc file requires knowledge of XML syntax. Do not attempt to change settings that already exist within the file.

1) Locate the version of the Designer.xdc file that is installed with Designer.

   *If Designer is installed separately from Workbench, the Designer.xdc file is in this location: [installation folder] \Designer*

   *If Designer is installed with Workbench, the Designer.xdc file is in this location: [installation folder]\Workbench\Designer*

2) Make a backup copy of the Designer.xdc file (good practice).
3) Open Designer.xdc by using a line editor or XML-capable editor.
4) In the `deviceInfo` element under the `Medium` heading, add a `medium` element for each of your custom medium (paper type).
5) Save the file as Designer.xdc in the same location as the original file.
6) Verify your addition to the Designer.xdc file by restarting Designer and then applying the new paper type (media type) to a form's master page.
7) Using XDC Editor, add the paper type to the XDC file for the target printer and associate the paper type to an input paper tray. (See Specifyingmedia and trays.)

   **NOTE:** Reinstalling or upgrading Designer overwrites your revised Designer.xdc file with the original version of the file. Store a backup copy of your revised file separate from the Designer installation.

## XDC files installed

Your installation includes several XDC files that describe the more common printers used. Modify the XDC files if your printers require different configuration or are not supported by the XDC files that are included with your installation. In this case, you can use XDC Editor to create or modify XDC files to meet your requirements.

As your starting point, modify copies of the XDC files that are available from the AEM forms SDK.

As your starting point, modify copies of the XDC files that are available from the Adobe document services SDK.

The SDK includes sample and template XDC files. The sample XDC files are for specific types of printers. Template XDC files are for generic print description languages such as PostScript language level 2 or PCL 5c. After you customize an XDC file for your printer, you can store the file in the Adobe document services server or on a local hard drive.

**NOTE:** Using a tool other than XDC Editor to modify XDC files is not recommended. Undetected format or syntax errors in the XDC file can cause print problems that are difficult to diagnose, and the cache for an XDC file is not dynamically updated after the file is modified.

### Template XDC files for generic print description languages

| Filename | Description |
|---|---|
| acrobat6.xdc | For use in rendering output in Adobe PDF 1.5. This file should not be modified in any way. Attempting to open this file in the XDC Editor results in an error message. |
| acrobat7.xdc | For use in rendering output in PDF 1.6. This file should not be modified in any way. Attempting to open this file in the XDC Editor results in an error message. |
| adobepdf.xdc | For use in rendering output in PDF 1.6 and 1.7. This file should not be modified in any way. Attempting to open this file in the XDC Editor results in an error message. |
| dpl203.xdc | Datamax 203 DPI printer; supports Datamax printer configuration file |
| dpl300.xdc | Datamax 300 DPI printer; supports Datamax printer configuration file |
| dpl406.xdc | Datamax 406 DPI printer; supports Datamax printer configuration file |
| dpl600.xdc | Datamax 600 DPI printer; supports Datamax printer configuration file |
| hppcl5c.xdc | Printers that support the HP® PCL 5c printer language (color) |
| hppcl5e.xdc | Printers that support the HP PCL 5e printer language (monochrome) |
| ipl203.xdc | Intermec label printer with 203 DPI resolution; supports Intermec Printer Language (IPL) |

| Filename | Description |
|---|---|
| ipl300.xdc | Intermec label printer with 300 DPI resolution; supports Intermec Printer Language (IPL) |
| ipl400.xdc | Intermec label printer with 400 DPI resolution; supports Intermec Printer Language (IPL) |
| ps_plain.xdc | Printers that support PostScript language level 2 (monochrome and color) |
| ps_plain_level3.xdc | Printers that support PostScript language level 3 (monochrome and color) |
| ps_plain_mt.xdc | Printers that support the PostScript printer language and use MT (Monotype) font names. This XDP file is no longer required but is included for compatibility with previous releases. (See SAP Note 867662.) |
| tpcl203.xdc | Toshiba TEC 203 DPI barcode label printer that supports Toshiba TEC Printer Command Language (TPCL) |
| tpcl305.xdc | Toshiba TEC 305 DPI barcode label printer that supports Toshiba TEC Printer Command Language (TPCL) |
| tpcl600.xdc | Toshiba TEC 600 DPI barcode label printer that supports Toshiba TEC Printer Command Language (TPCL) |
| zpl203.xdc | Zebra label printers (monochrome) at 203 dpi (8 dots/mm) |
| zpl300.xdc | Zebra label printers (monochrome) at 300 dpi (12 dots/mm) |
| zpl600.xdc | Zebra label printers (monochrome) at 600 dpi (24 dots/mm) |

| Filename | Description |
|---|---|
| dpl203.xdc | Datamax 203 DPI printer; supports Datamax printer configuration file |
| dpl300.xdc | Datamax 300 DPI printer; supports Datamax printer configuration file |
| dpl406.xdc | Datamax 406 DPI printer; supports Datamax printer configuration file |
| dpl600.xdc | Datamax 600 DPI printer; supports Datamax printer configuration file |
| hppcl5c.xdc | Printers that support the HP PCL 5c printer language (color) |
| hppcl5e.xdc | Printers that support the HP PCL 5e printer language (monochrome) |
| ipl203.xdc | Intermec label printer with 203 DPI resolution; supports Intermec Printer Language (IPL) |
| ipl300.xdc | Intermec label printer with 300 DPI resolution; supports Intermec Printer Language (IPL) |
| ipl400.xdc | Intermec label printer with 400 DPI resolution; supports Intermec Printer Language (IPL) |

| Filename | Description |
|---|---|
| ps_plain.xdc | Printers that support PostScript language level 2 (monochrome and color) |
| ps_plain_level3.xdc | Printers that support PostScript language level 3 (monochrome and color) |
| ps_plain_mt.xdc | Printers that support the PostScript printer language and use MT (Monotype) font names. This XDP file is no longer required but is included for compatibility with previous releases. |
| tpcl203.xdc | Toshiba TEC 203 DPI barcode label printer that supports Toshiba TEC Printer Command Language (TPCL) |
| tpcl305.xdc | Toshiba TEC 305 DPI barcode label printer that supports Toshiba TEC Printer Command Language (TPCL) |
| tpcl600.xdc | Toshiba TEC 600 DPI barcode label printer that supports Toshiba TEC Printer Command Language (TPCL) |
| zpl203.xdc | Zebra label printers (monochrome) at 203 dpi (8 dots/mm) |
| zpl300.xdc | Zebra label printers (monochrome) at 300 dpi (12 dots/mm) |
| zpl600.xdc | Zebra label printers (monochrome) at 600 dpi (24 dots/mm) |

**Sample XDC files for use with specific printers**

The files in this table are sample XDC files that support the features of specific printers, such as resident fonts, paper trays, and staplers. The purpose of these samples is to help you understand how to set up your own printers by using device profiles.

| Filename | Description |
|---|---|
| hp4350pcl5e.xdc | HP 4350 printer device profile using PCL5e (monochrome) |
| hp4350ps.xdc | HP 4350 printer device profile using PostScript (monochrome and color) |
| lmt644pcl5e.xdc | Lexmark T644 printer device profile using PCL5e (monochrome) |
| lmt644ps.xdc | Lexmark T644 printer device profile using PostScript (monochrome and color) |

After you modify an XDC file, use the administration report for XDC files RSPO0022 to manage the mapping of SAP device types to XDC files. For information about this report, search the SAP Help portal for “Administering XDC Files for SAP Device Types (Report RSPO0022)”.

After you modify an XDC file, reference it as part of developing a client application. For information about programmatically referencing an XDC file, see Programming with AEM forms

. For information about referencing an XDC file in a process, see Workbench Help

.

### XDC file storage

Store XDC files in the SAP database.

If you store XDC files on your local computer, deploy them to Adobe document services, as described in DeployingXDC files.

Store XDC files in AEM forms applications or on the computer hosting Workbench. Here are some advantages to storing XDC files in the AEM forms applications:

- XDC files are available to Output.
- XDC files are available to Workbench users on other computers.
- XDC files stored in the AEM forms applications are automatically assigned a version number and timestamp.

If you store XDC files on your local computer, deploy them to Output, as described in DeployingXDC files

## 20.2. Creating and modifying XDC files

XDC Editor guides you through the process of creating and modifying XDC files that Adobe document services uses when printing a form. Adobe document services uses the information in such files to set up and control the printer and to position characters on the printed page.

XDC Editor guides you through the process of creating and modifying XDC files that Output uses when printing a form. Output uses the information in such files to set up and control the printer and to position characters on the printed page.

### Before you begin

Before you create or modify XDC files, obtain sample and template XDC files to use as your starting point. You should also have your printer's reference manual.

#### Finding the XDC files installed with SAP NetWeaver Application Server

If Adobe document services are deployed to the SAP NetWeaver Application Server, the XDC files are located in this directory:

/<*DIR_GLOBAL*>/AdobeDocumentServices/lib

#### Copying sample and template XDC files to your local hard drive

When creating or modifying an XDC file, use an existing XDC file as a starting point. That is, you cannot create an XDC file from scratch. If you are just getting started with XDC Editor, use the XDC files installed with the AEM forms server SDK.

1) *[dep root]*\Workbench\sdk\samples\SamplesConfig\bin
2) Find the XDC samples located in this Workbench directory:
   *[dep root]\ Workbench\sdk\samples\Output\IVS\XDC*

For more information about the BAT file and running it to configure the samples, see the Readme file in the *[dep root]*\Workbench\sdk\samples\SamplesConfig directory.

### Finding documentation for your printer

If you do not have the documentation for your printers, you can get it by searching the manufacturer's websites for documentation about your particular model type. Here are some tips on finding documentation from more familiar manufacturers:

**Hewlett-Packard**

Download printer documentation from www.hp.com

or go to HP Manuals

for a list of all Hewlett-Packard printer manuals.

**Ricoh**

Download printer documentation from www.ricoh-usa.com/downloads

.

**Lexmark printers**

Download printer documentation from www.lexmark.com/publications/techref.html

.

**Datamax label printers**

Download printer documentation from www.datamaxcorp.com/software/

**Intermec label printers**

Download printer documentation from http://www.intermec.com/support/manuals/index.aspx

**Toshiba TEC label printers**

Download printer documentation from www.toshibatecusa.com/ServicesandSupport/Owner-Manual.aspx

If you still cannot find your printer's reference manual, this guide provides alternate ways of finding the information you need.

## Copying XDC files into your application

AEM forms server SDK provides sample XDC files for PostScript (PS), PCL printers, and label printers such as Datamax, Intermec, Toshiba TEC, and Zebra. You can copy the required files into your applications so that you can deploy them to the server. Do one of the following tasks:

- Drag the required files from your local file system into the application.
- Copy the required files from your local file system and then paste them into your application.

You can check the files into the server so that they become available to the AEM forms server. Until you check them in, the files reside inside your application, but on the local server. Files that are added to an application have an icon with a green plus (+) sign.

## Opening an XDC file

In Workbench, you can open and modify your own XDC files. You can also open template XDC files that you use as starting points for creating your own new XDC files. You must select an XDC file that uses the same PDL as the file you are creating.

In Workbench, you cannot create an XDC file from scratch. Instead, open another XDC file as a template. You can use the sample and template XDC files that are installed with AEM forms server or other XDC files that you created as templates. After you modify the settings in the template, XDC Editor ensures that you provide a unique name for the new XDC file.

If the version of the XDC file on the server has changed, synchronize the file with the server and then open the updated file.

### To open a local XDC file

1) In the Workbench toolbar, select File > Open File.
2) Navigate to the folder in your local file system that contains the XDC file and click Open.

### To open an XDC file in an application

1) Navigate to the folder in your application that contains the XDC file.
2) From the Applications view, right-click the XDC file and select Check Out. The checked-out file's icon changes to one with a green check mark, indicating that it is checked out.
3) Double-click the file to open it.

**NOTE:** If you open a file without checking it out first, Workbench prompts you to confirm if you want to check out the file first when you make a change.

## Opening an XDC file in Eclipse

In Eclipse, you can open and modify your own XDC files.

1) In the Eclipse toolbar, select File > Open.
2) Perform one of the following tasks:
   – If you are modifying an existing XDC file, select the XDC file to modify.
   – If you are creating a new XDC file, select File > New > Other > Adobe XDC Editor . Select an XDC file that uses the same PDL as the file you are creating.

   *In SAP NetWeaver 7.32, copy the XDC file from the /<DIR_GLOBAL>/AdobeDocumentServices/ lib folder to a folder that is accessible from XDC Editor. When you finish modifying the file, copy the updated or new XDC files back to the server.*

*In SAP NetWeaver 7.1, use the Adobe document services configuration pages in the SAP NetWeaver Administrator to upload and download the XDC files.*

**TIP:** When creating an XDC file for an HP or Lexmark printer, start from a sample XDC file instead of a generic XDC file. The sample XDC files contain sets of HP or Lexmark printer font descriptions. For a description of the XDC files in the samples folder, see XDCfiles installed.

3) Modify the XDC file to match your target printer.
4) Select File > Save and provide a new name that uniquely identifies the XDC file.

   *For information about setting characteristics in your XDC file, see Specifying mediaand trays, Specifyingfont descriptions, Specifyingprinter capabilities, and Specifyingsequences.*

## Specifying a name for the XDC file

You can specify a human-readable name for an XDC file. The Designer Print dialog box displays this name next to the XDC file name, which makes it easier for users to select the correct XDC file to use to print a document.

1) Select and open an XDC file. (See Openingan XDC file.)
2) Select and open an XDC file. (See Openingan XDC file in Eclipse.)
3) In the XDC Name field, enter a human-readable name.
4) Save this file, and then check it into the server. (See DeployingXDC files.)
5) Export and copy the modified XDC file to the Designer installation folder. (See DeployingXDC files.)

## Specifying media and trays

Using XDC Editor, you can define printer input trays and media types separately, and then map them. This separation allows you to use the media types that are separately defined by designers who laid out forms using Designer. Typically, you define the media types, fonts, sequences, and printer capabilities, and then map the media types to trays. However, because the media types are standardized and usually form authors define the media types that they intend to use with the forms they author, generally, you do not need to add or edit media types. In addition, because the mapping between media types and trays require exact names of media types, it is recommended that you use the media types that the form designer makes available.

You can map media types (paper types) that are available on the printer to input trays that those types are loaded in. You can also specify the output trays that are available on the printer. Here, you can select the tray and media types by the names you assigned in Designer, and then map media types and trays.

### Specifying media types

Typically, form authors using Designer specify the media type that each page in the form is printed on, choosing from the media types specified in the Designer XDP file (Designer.xdp). When Adobe document services prints the form, it uses the XDC file for the printer to associate the media type for a page with a specific input tray. It also uses the XDC file to determine the printer-specific commands for selecting the input and output trays. Adobe document services incorporates these commands into the setup instructions it sends to the printer.

Typically, form authors using Designer specify the media type that each page in the form is printed on, choosing from the media types specified in the Designer XDP file (Designer.xdp). When Output prints the form, it uses the XDC file for the printer to associate the media type for a page with a specific input tray. It also uses the XDC file to determine the printer-specific commands for selecting the input and output trays. Output incorporates these commands into the setup instructions it sends to the printer.

XDC Editor allows you to specify the media types that form authors define for use in their master pages. Within the XDC Editor, you can independently define the media types by adding, removing, or editing media types. Typically, you use the same name as the media type that the form author used, and then define the characteristics of each media type. You can then map the input trays to media types.

The XDC file for the target printer maps media types to input trays. To print a form that uses non-standard media, you may have to modify that printer's XDC file to specify the input tray that the media is located in. To do this, modify your XDC file or XCI properties so that the printer feeds paper from the correct input tray. In some cases, you may have to modify both the XDC file and the XCI properties.

Follow the scenario below to change the type of paper that is specified in the form or if the form does not have a type of paper specified.

Modifying the XDC file to specify media and tray selection involves the following general tasks:

Determiningor specifying the media used by a form

Identifyingyour printer's input tray numbers

Mediaand tray mapping (PCL and PostScript Printers only)

DeployingXDC files

### Determining or specifying the media used by a form

You can determine or specify the medium used in a blank form by using Designer.

*Determine or specify the medium used in a blank form*

1) Open a form in Designer.
2) Select Window > Hierarchy to display the Hierarchy palette.
3) In the Hierarchy palette, select one of the pages subordinate to the Master Pages entry (for example, select form 1 > (Master Pages) > Page 1).
4) Show the Object palette by selecting Window > Object. The Paper Type list on the Master Page tab displays the currently selected paper type for the blank form. (*Paper type* and *media type* are synonyms.)

   **NOTE:** A blank form can have multiple master pages, each of which has a different medium selection.
5) (Optional) In the Paper Type list, select a new media for the master page. The paper types available are derived from the XDC file that is used by Designer. For example, Designer presents a paper type of Letter Head derived from the XDC entry letterHead.

*Define media types in XDC Editor*

1) In Workbench, open the XDC file to modify.
2) In XDC Editor, open the XDC file to modify.

3) On the Mediums tab, click Add.
4) In the Add New Medium dialog box, do the following:
   – From the Stock pop-up menu, select the media type that the form author may have defined.
   – In the Long Edge field, specify the long-edge dimension of the media, appending the unit of measurement. Units can be expressed with three fractional digits (for example, 432 pt and 44.555 in). Supply this value even if the name of the media type implies its value.
   – In the Short Edge field, specify the short-edge dimension of the media, appending the unit of measurement. Supply this value even if the name of the media type implies its value.
   – (Printers except PostScript printers) In the Imaging Bounding Box field, specify the coordinates that define the printable area of the media. Provide one set of coordinates to define the lower-left points and another set of coordinates to define the upper-right points. Separate the values with commas and provide the unit of measurement with the values. For example, 0pt, 1pt, 288pt, 431pt describes a printing area with a lower-left corner that has the coordinates (0pt, 1pt) and an upper-right corner that has the coordinates (288pt, 431pt). For information about specifying the printable area, see your printer's manual. (See Findingdocumentation for your printer.)

### Remove a medium

You can remove a medium from the XDC file using XDC Editor. For PCL and PostScript supported printers, removing a medium from the XDC file removes tray-medium mapping that uses the selected medium.

*To remove a medium from the XDC file*

1) In Workbench, open the XDC file to modify.
2) In XDC Editor, open the XDC file to modify.
3) On the Mediums tab, select the medium to remove, and then click Remove.

### Modify a medium (PCL and PostScript printers only)

When you modify a medium, Tray-Medium mapping that uses the changed medium is also modified.

1) In Workbench, open the XDC file to modify.
2) In XDC Editor, open the XDC file to modify.
3) On the Mediums tab, select the medium to modify, and click Change.
4) In the Change Medium dialog box, change the relevant parameters, and click OK.

### Identifying your printer's input tray numbers

Now that you know the media types the master pages use in the blank form, identify the tray number that delivers each media type. The tray number may differ from the numbers printed or embossed on the trays themselves. Use these tray numbers when you modify the XDC file. Ideally, you should obtain tray information from the printer's manual; however, this Help describes other ways to obtain tray information for your printer.

*Obtaining information from your printer's manual*

If you have a copy of your printer's reference manual (Findingdocumentation for your printer), it should provide a list of possible input tray numbers. For example, the technical reference (not the user reference) for the Lexmark T632 printer shows the tray numbers in the table below, depending on whether PostScript or PCL emulation is selected. (See Specifyingprinter capabilities). Keep in mind that you can specify the trays only for PCL or PostScript printers.

**NOTE:** PostScript language level 2 uses tray names instead of tray numbers. Therefore, use tray names to identify input trays. (See Mediaand tray mapping (PCL and PostScript Printers only).)

| Input tray number using PCL emulation | Input tray number using PostScript emulation (language level 3 only) | Tray description |
|---|---|---|
| 0 | Not specified | Active source or eject page |
| 1 | 0 | Tray 1 (default) |
| 2 | 3 | Manual paper feed |
| 3 | 4 | Manual envelope feed |
| 4 | 1 | Tray 2 |
| 5 | 5 | Tray 3 |
| 6 | Not specified | Optional envelope feeder |
| 7 | Not specified | Auto select |
| 8 | 7 | Multipurpose feeder |
| Not specified | 2 | Multipurpose feeder or envelope feeder |
| 20 | 8 | Tray 4 |
| 21 | 9 | Tray 5 |
| 62 | Not specified | Optional paper source |

*Obtaining information from your printer*

Many printers provide an interface that lets you obtain information directly from the printer:

**Display panel or screen**

Displays information about the printer configuration or lets you request the printer to print a configuration page. Information may include tray numbers. In the case of networked printers, the configuration page may include an IP address.

### Web interface

Provides a user interface so that you can examine the printer configuration by using your web browser.

*Obtaining information from your printer's PPD file (PostScript printers)*

For PostScript printers, if you cannot obtain the printer's reference manual or information from the printer itself, you can obtain tray information from the printer's PPD (PostScript printer description) file. This applies only to printers that support the PostScript language. You can download PPDs from the Adobe Printer Drivers

site. Alternatively, you can search the Internet for other PPD download websites or go to CommonUNIX Printing System (CUPS)

.

The following lines come from the PPD file for the Ricoh cl 3000 printer:

```
*InputSlot MultiTray/Bypass Tray: "<</MediaPosition 0>> setpagedevice"
*InputSlot 1Tray/Tray 1: "<</MediaPosition 1>> setpagedevice"
*InputSlot 2Tray/Tray 2: "<</MediaPosition 2>> setpagedevice"
*InputSlot 3Tray/Tray 3: "<</MediaPosition 3>> setpagedevice"
```

In this case, the text string `2Tray/Tray2` that appears before the colon (":") corresponds to the actual printer tray. `<<MediaPosition 2>>` is the PostScript command to select that input tray. The input tray number is given as a parameter of `MediaPosition`; therefore, the input tray number to be used is "2".

### Trays (PCL and PostScript printers only)

The Tray panel in the XDC Editor allows you to manage trays that are available in your printer and then map them to specific media types that are available from the Stock menu.

*Add a tray*

1) In the XDC Editor, open the PCL or PostScript XDC file that you want to edit.
2) In the Tray panel, click Add.
3) In the Add New Tray dialog box, do the following, and click OK:
   - In the Input Tray Name field, enter a unique, meaningful name for the input tray.
   - (PostScript language level 3 printers) In the Input Tray Number field, specify the tray number that identifies the input tray.
   - In the Tray Type field, specify the PostScript media type that is associated with a particular input tray. Tray Type values are either predefined by the printer's manufacturer or configured by the system administrators by using the printer's front panel. Examples of predefined Media Type values are PLAIN, COATED, and GLOSSY.
   - (PostScript language level 2 printers) Omit values from the Tray Number field and specify the input tray type in the Input Tray Type field.

*Edit Tray details*

When you rename an input tray, for PCL and PostScript printers, Tray-Medium mapping that uses the edited tray is also updated.

1) In Workbench, open the XDC file that you want to modify.
2) In XDC Editor, open the XDC file that you want to modify.
3) On the Trays tab, select the tray that you want to edit.
4) In the Change Input Tray dialog box, edit the parameters and click OK to save them.

### Media and tray mapping (PCL and PostScript Printers only)

For printers that support PostScript or PCL languages, you can map the available media types and trays. Keep in mind that you are mapping the media types to Tray Names, not tray numbers. In the Tray panel, you can add, change, or delete input tray characteristics. Before you begin the procedures, ensure that your XDC file is configured for your printer, and identify the input tray numbers and tray names for your printer. (See Specifyingprinter capabilities and Identifyingyour printer's input tray numbers.)

*Map a medium to a tray*

1) At the bottom of the XDC Editor panel, click the Trays tab.
2) Click Add to the right of the Medium To Tray Mappings table.
3) From the Stock menu, select the media type you are want to map to a tray.

   *The media types that are available from the Stock menu reflect the media types that are predefined in Designer. Using Designer, form authors specify the paper types (media types) for the master pages that are used on a form, choosing from a list of the same predefined media types (paper types). (See Determiningor specifying the media used by a form.)*

   *Notice that some of the entries in the Stock column are not preceded by an asterisk. These entries reflect a subset of the media types that are predefined in Designer. Entries in the Stock column that are preceded by an asterisk are defined only in the XDC file being edited.*

   *If you add a media type that is not already defined in the Designer.xdc file, add the new media type to the copy of the Designer.xdc file that Designer uses and then restart Designer. Thereafter, form authors can select the newly created stock name in the Master Page object property. (See Adding-new paper types to the Designer.xdc file.)*
4) In the Input Tray Name field, select media name that you want to map to the tray that you selected already, and click OK.

*Delete a media and tray mapping entry*

1) At the bottom of the XDC Editor panel, click the Trays tab.
2) Select the entry to delete from the mapping table, and click Remove.

**NOTE:** If you delete a medium that was used in any of the tray-medium mapping, the mapping is removed when the medium is removed.

### (PostScript- and PCL-based XDC files) Adding or changing output trays

In the Output Tray panel, you can add, change, or delete output tray characteristics.

*Add an output tray*

1) At the bottom of the XDC Editor panel, click the Medium and Trays tab, and then click Add.
2) In the Output Tray Name field, specify a name that uniquely identifies the output tray. This name can be referenced by API parameters and by Adobe document services parameters.
3) In the Output Tray Name field, specify a name that uniquely identifies the output tray. This name can be referenced by API parameters and by Output parameters.
4) In the Tray Bin field, specify the output bin number. This value corresponds to the Output Bin entry in the Printed Output Options dialog box of the Output Bin specified in the form design.
5) In the Tray Bin field, specify the output bin number. This value corresponds to the Output Bin entry in the Printed Output Options dialog box of the generated Printed Output service of Output.
6) In the Tray Number field, specify the tray number. This number can be referenced by API parameters and by Adobe document services parameters.
7) In the Tray Number field, specify the tray number. This number can be referenced by API parameters and by Output parameters.

*Modify an output tray*

1) At the bottom of the XDC Editor panel, click the Medium and Trays tab.
2) Select the entry to modify and click Modify, or double-click the entry to modify.
3) Modify the necessary fields. (See Toadd an output tray.)

## Specifying font descriptions

You must provide certain font descriptions in your XDC files. Adding font descriptions involves these tasks:

1) Preparingto add fonts
2) Addfonts

**NOTE:** XDC Editor enables definition of fonts that support any one of these character encodings: ISO-8859-1, ISO-8859-2, Shift-JIS, KSC-5601, GBK, BIG-5, and character-specific. Symbol encodings are used for symbolic fonts, such as Wingdings and ITC Zapf Dingbats.

### Preparing to add fonts

The XDC file must specify the typefaces for each printer-resident font that can appear in forms that are printed on that printer, including fonts that are installed in the printer on a device such as a cartridge, SIMM, or DIMM. A *typeface* is a coordinated set of glyphs that are designed with a stylistic unity.

Here are the general tasks for specifying printer-resident fonts:

1) Determinewhich fonts are resident on a printer
2) Designa form to use printer-resident fonts

3) Identifyescape sequences that specify printer-resident fonts (PCL only)

**NOTE:** Your right to use fonts provided by parties other than Adobe is governed by the license agreements provided to you by such parties in connection with those fonts, and is not covered under your license to use Adobe software. Adobe recommends that you review and ensure that you are in compliance with all applicable non-Adobe license agreements before using non-Adobe fonts with Adobe software, particularly with respect to use of fonts in a server environment.

*Determine which fonts are resident on a printer*

Your printer's technical reference manual should have information about printer-resident fonts. (See Findingdocumentation for your printer.) If you cannot obtain the manual, you may obtain information from the printer itself:

- Use the front panel on the printer to print a list of fonts.
- If the printer has a built-in web server, use your web browser to request the printer to print a list of fonts.

*Design a form to use printer-resident fonts*

Using Designer, you can specify the name of printer-resident fonts by entering the name in the Font palette. Usually, a font is selected from the drop-down list; however, in Designer, you can enter a font name. The name you enter must be identical to the font name that is identified in Determinewhich fonts are resident on a printer.

You can also use the SAP Cascading Font configuration to map a font that is used in the templates to the printer-resident font on this printer. For more information about using the SAP Cascading Font configuration, see the CSN note *Adobe document services Font Use*.

*Identify escape sequences that specify printer-resident fonts (PCL only)*

PCL printers use escape sequences to specify printer-resident fonts. These codes specify characteristics such as the font name, size, and angle. XDC Editor can usually determine these escape sequences; however, situations may occur where it does not have adequate information, such as in the following situations:

- The font resides only on the printer, such as magnetic ink character recognition (MICR) fonts.
- The font uses a symbol set other than those that XDC Editor supports, which include OpenType® and TrueType fonts that support any one of these character encodings: ISO-8859-1, ISO-8859-2, Shift-JIS, KSC-5601, GBK, BIG-5, and character-specific. Symbol encodings are used for symbolic fonts, such as Wingdings and ITC Zapf Dingbats.

If XDC Editor cannot determine the escape sequence for the printer-resident font that a form uses, enter it in the Escape Sequences field. (See DeterminingPCL font sequences.)

### Add fonts

1) Select and open an XDC file. (See Openingan XDC file.)
2) Select and open an XDC file. (See Openingan XDC file in Eclipse.)
3) At the bottom of the XDC Editor panel, click the Fonts tab.

4) In the Font panel, click Add.
5) Select a font to add. If the font resides in the Microsoft® Windows® system folder (Windows/Fonts), create a temporary folder elsewhere on your local hard drive and copy the font into that folder. Windows prevents XDC Editor from accessing fonts in the Windows system folder.
6) Double-click the font file to add it to the list of fonts in the Font panel.

   *If the font you selected in step 4 supports multiple character encodings, XDC Editor displays a dialog box where you can select the encodings to use.*
7) Select the encodings that apply. For each encoding you specify, XDC Editor creates a typeface entry in the Font table. Apply the remaining steps to each of the steps for those typeface entries.
8) (PCL XDC files) Ensure that Edit Sequence As String is selected, and click Generate Sequence. If XDC Editor recognizes the font, it creates a font sequence for you that omits the printer-specific typeface code for the font. Notice that XDC Editor fills all Font Sequence fields except the Typeface Code field.
9) (PCL XDC files) If XDC Editor created a font sequence for you, deselect Edit Sequence As String, enter the printer-specific typeface code in the Typeface Code field, and then click Apply.

   *For example, the typeface code for Courier is 4099. You can find the typeface codes in your printer manual or in the printer display panel. (See Obtaining informationfrom your printer's manual and Obtaininginformation from your printer).*
10) (PCL XDC files) If XDC Editor did not create a font sequence, supply one in the field under the Edit Sequence as String label (see DeterminingPCL font sequences), and then click Apply.
11) (PostScript and ZPL XDC files) Enter the printer's name for the font in the Font Sequence field. For PostScript files, the printer's name for the font is a combination of the typeface name, weight, and angle. For ZPL files, the printer's name for a font is one or more uppercase letters. You may find these names in these sources:
    - Your printer manual (see Obtaininginformation from your printer's manual)
    - Information page printed by your printer
    - Printer display panel (see Obtaininginformation from your printer)

**Remove a font**

1) Select and open an XDC file. (See Openingan XDC file.)
2) Select and open an XDC file. (See Openingan XDC file in Eclipse.)
3) At the bottom of the XDC Editor panel, click the Fonts tab.
4) In the Font panel, select the font and click Remove.

For information about installing fonts and ensuring consistent appearance of fonts as viewed by a form author and as printed by a network printer, see Adobedocument services - Configuration

on the SAP Help Portal.

**Font sequences for label printers**

For label printers, the font sequences are defined in the XDC files. For example, for a Datamax printer, each of the Datamax fonts have the sequences defined in the XDC file as follows:

```
<seq id="Datamax0 2.5 point_Normal_Normal_ISO-8859-1">0,0,1,1,0</seq>
```

| Sequence | Meaning |
|---|---|
| 0 (First value) | Indicates the font type. 0: bitmap font; 1: scalable font. |
| 0 (Second value) | Specifies the font number of the printer-resident font. |
| 1 (Third value) | Specifies the font width magnification (applicable for bitmap fonts only). |
| 1 (Fourth value) | Specifies the font height magnification (applicable for bitmap fonts only). |
| 0 (fifth value) | Specifies vertical text offset to be applied.<br>***NOTE:*** *Baseline shift is not applicable for Toshiba TEC printers.* |

For the scalable fonts, the width and height magnifications will be automatically calculated depending on the point size. The bitmap fonts have the point size mentioned in their name. Use this point size at the time of designing the form.

## Specifying printer capabilities

On the Printer Capabilities tab, you can specify the printer description language. For printers that support the Zebra printer language, you can also specify settings for printing radio frequency identification (RFID) labels and tags.

### Specifying the language supported by a printer

All XDC files must specify the language that the target printer supports. In general, this language is predefined in the XDC file that you use as a starting point for your XDC file.

*Configure device options*

1) Select and open an XDC file. (See Openingan XDC file.)
2) Select and open an XDC file. (See Openingan XDC file in Eclipse.)
3) At the bottom of the XDC Editor panel, click the Printer Capabilities tab.
4) For an XDC file that is based on the PCL language, specify the language and the language level. For an XDC file that is based on the PostScript language, specify the manufacturer, model, language, and language level.

   *For example, the hp4530ps.xdc file specifies the Manufacturer as HP, the Model as HP LaserJet 4350PS, the Language as ps, and the Language Level as 3. Change any of the preset values to match the physical characteristics of your printer.*
5) (PCL supported printers) Specify whether the printer supports logical operations. Select true if the printer supports the Logical Operations command that applies logical functions such as AND, OR, and XOR to the basic PCL print model operations.
6) (PostScript supported printers) Specify the encoding formats that are used for font encoding. To access characters that are not in StandardEncoding in the PostScript language, applications must re-encode fonts.

7) Save your changes.

### Specifying RFID characteristics (for printers that support ZPL only)

On the Printer Capabilities tab, you can specify settings that are used with RFID printers/encoders. Such RFID printers/encoders simultaneously encode an integrated circuit chip and print barcodes and text. The encoding is accomplished by using radio frequency waves.

RFID stands for *radio frequency identification*. It is an automatic identification technology whereby digital data that is encoded in an RFID tag or *smart label* is captured by a reader by using radio waves. That is, RFID is similar to barcode technology but uses radio waves to capture data from tags rather than optically scanning the barcodes on a label. RFID does not require the tag or label to be visible in order to read its stored data.

Preparing to print forms to an RFID printer involves the following:

- Verifying that your form specifies barcodes that support RFID
- Obtaining information about the RFID labels and tags being encoded
- Obtaining characteristics about the RFID printer/encoder.

To verify that your form specifies barcodes that support RFID, see *Designing Forms for AEM forms output* .This document also provides guidance on designing a form for printing to an RFID printer or encoder.

*Determine characteristics of RFID labels or tags*

1) Identify the brand or standard of the RFID labels or tags. Examples of tag brands include Texas Instruments Tag-it, Phillips I•Code, and Inside Technologies Picotag 2K. Examples of tag standards include EPC Class 0, EPC Class 1 96-bit, and ISO 15693.
2) Determine the distance (in vertical dot-rows) between the RFID transponder (the antenna and microchip embedded in the label or tag) and the leading edge of the label or tag.

*Determine characteristics of the target RFID printer/encoder*

1) Obtain a copy of your printer manual. (See Findingdocumentation for your printer.)
2) Verify that your printer supports RFID printing and the RFID labels or tags you plan to print.
3) Determine the integer for the tag type you plan to print. Typically, this information appears in the printer manual in the description of RFID Setup, which corresponds to the ZPL command `^RS`.
4) If possible, verify your RFID settings by using the printer's LCD display to initiate a test print.

   *Some RFID printers/encoders have an LCD display that lets you test your ability to write and read to the RFID transponder. Passing this test means that your printer is set up correctly and you are ready to print/encode RFID label or tags.*

*Specify print setup for an RFID printer/encoder*

1) Select and open an XDC file. (See Openingan XDC file.)
2) Select and open an XDC file. (See Openingan XDC file in Eclipse.)
3) At the bottom of the XDC Editor panel, click the Printer Capabilities tab.

4) Select the RFID block retries field and enter the number of read/write retries before the printer declares the transponder defective. This value must be an integer, typically between 0 and 10.
5) Select the RFID label retries and enter the number of failed labels or tags that cause the print job to be terminated. The accepted values for this option is between 0 to 10.
6) Select the RFID tag type and enter the printer-specific digit for the label or tag type that is installed in the printer. (See Determinecharacteristics of the target RFID printer/encoder.)
7) Select the RFID transponder position and enter the distance (in vertical dot-rows) between the RFID transponder and the leading edge of the label (see Determinecharacteristics of RFID labels or tags). Accepted values are between 0 to the length of the label. If the transponder is located on the leading edge of the label, set this option to 0.

## Specifying sequences

Using XDC Editor, you can specify PDL statements for a set of predefined sequence names. The sequences are bits of printer programming languages that are represented as XML. Before you attempt to modify a sequence, be aware of the associated PDL sequence conventions.

### PDL sequences

PDL sequences are short PDL sequences that Adobe document services injects into the PDL syntax that is sent to the printer at specific points in the print job, such as at the beginning of a page or a record. Adobe document services adds the PDL sequences to perform events, such as starting a print job, starting a new document within a print job, or simply drawing a line.

PDL sequences are short PDL sequences that Output injects into the PDL syntax that is sent to the printer at specific points in the print job, such as at the beginning of a page or a record. Output adds the PDL sequences to perform events, such as starting a print job, starting a new document within a print job, or simply drawing a line.

Each sequence is associated with a name that uniquely identifies the sequence. XDC Editor lets you modify the sequences that are associated with predefined names. (See Predefinedvariables.)

### Predefined variables

The XDC files include several predefined variables, such as sequence names and printer-specific values.

*Sequence names*

The XDC files that are delivered with your installation contain predefined sequences. If required, you can use XDC Editor to modify these sequences. The names defined for PostScript and PCL XDC files differ from the names defined for label printer XDC files.

This table describes the predefined sequence names for PostScript and PCL XDC files.

| Name | PostScript | PCL | When a sequence is invoked |
|---|---|---|---|
| preDoc | Yes | Yes | To set up the printer for the document. The sequence is invoked after the preamble sequence but before the preRecord sequence. XDC Editor does not enable editing the preamble sequence. |
| postDoc | Yes | Yes | To set up the printer after the document. The sequence is invoked after the postRecord sequence but before the postamble sequence. |
| preRecord | Yes | Yes | To set up the printer before every record. A *record* is a repeating set of data. Typically, a record holds the data from a single form instance. |
| postRecord | Yes | Yes | To set up the printer after every record. |
| prePage | Yes | Yes | To set up the printer before every page. |
| postPage | Yes | Yes | To set up the printer after every page. |
| paginationSimplex | Yes | Yes | To set up long-edge duplex. The result of this setting is to impose each consecutive pair of pages to enable binding on the long edge. |
| paginationDuplex ShortEdge | Yes | Yes | To set up short-edge duplex. The result of this setting is to impose each consecutive pair of pages to enable binding on the short edge. |
| pagination | Yes | N/A | To set up the printer pagination feature, such as duplex. |
| stapleOn | Yes | Yes | To enable the printer's staple feature. |
| stapleOff | Yes | Yes | To disable the printer's staple feature. |
| staple | Yes | N/A | To staple the document. |
| jogNone | Yes | Yes | To disable the printer's jog feature. |
| jogPageSet | Yes | Yes | To enable the printer's jog feature. |
| jog | Yes | N/A | To jog the tray. |

This table describes the predefined sequence names for Zebra XDC files.

| Name | When a sequence is invoked |
|---|---|
| startDoc | Not applicable. |
| endDoc | Not applicable. |

| Name | When a sequence is invoked |
|---|---|
| startPage | To set up the printer at the start of each page. |
| endPage | To set up the printer at the end of each page. |
| carat | To specify the name of the instruction prefix character. Most ZPL commands are prefixed by the caret (^) character, such as ^A, ^B1, ^B2, and so on.<br>Some older Zebra printer models may erroneously interpret the default ^ character as the second byte of a double-byte Japanese character. This case is the only one in which you should need to change the default ^ character.<br>You can change this character to another character, such as the exclamation mark (!). The ^ character is substituted with the character you specify, such as !A, !B1, !B2, and so on. |
| tilde | To execute control instructions that cause the printer to take a specific action immediately, such as clearing the memory or feeding a blank label. |
| copy | To specify the number of copies. |

*Variables resolved*

XDC files for PRC printers use variables to specify font size or font pitch typeface (font) characteristics. These variables are described in the Identifyescape sequences that specify printer-resident fonts (PCL only) section.

**Use of variables in sequences**

A sequence can use `var` elements that have corresponding XCI settings. At run time, Adobe document services substitutes the variable in the XDC file with the value specified in the equivalent XCI setting. Examples of variables that map to XCI settings include those in this table:

A sequence can use `var` elements that have corresponding XCI settings. Developers can specify these XCI settings in a process or by using the Output API. At run time, the Output server substitutes the variable in the XDC file with the value specified in the equivalent XCI setting. Examples of variables that map to XCI settings include those in this table:

| Sequence name | Variable name | Equivalent XCI setting |
|---|---|---|
| copy | `<var name="numCopies" />` | setCopies |
| pagination | `<var name="duplex" />` | setPagination |
| staple | `<var name="staple" />` | setStaple |
| jog | `<var name="jog" />` | setOutputJog |

A sequence can use `var` elements to reference named sequences that are defined earlier in the XDC file or defined by the PDL. The following example uses the `pageWidth` and `pageHeight` variables:

```
<var name="pageWidth"/> <var name="pageHeight"/>
```

A sequence can also reference other sequences by name. Such references are expressed as an XML segment that defines a `seq` element. The following example is of a `startDoc` sequence from ps_plain.xdc:

```
<seq use="#pa_preamble" />&#xa;<seq use="#textEncodings" />
```

The above sequence references two other sequences (`pa_preamble` and `textEncodings`), which are defined elsewhere in the XDC file. XDC Editor does not enable modifications of these sequences or any other sequences that are not visible in the Sequences tab.

For more examples of sequences, see the XDC files provided with your installation.

### Encoding PCL sequences

PDL commands that are entered for a sequence must conform to XML conventions. As a result, you must use the `<ESC/>` element and XML escape sequences to represent characters that have special meaning in XML. The following table shows the more common escape sequences.

| Character | Escape sequence |
|---|---|
| Escape | `<ESC/>` |
| Ampersand (&) | `&` |
| Left angle-bracket (<) | `<` |
| Right angle-bracket (>) | `>` |
| Line feed | `
`<br>`
` |
| Carriage return | `` |

The following example is of a `preDoc` sequence for a PCL printer. Before this sequence is included in the PCL stream that is sent to the printer, the escape sequences are replaced by the characters they represent. The escape characters in this sequence appear in bold:

```
<ESC/>%-12345X@PJL RDYMSG DISPLAY=""&#13;&#10;@PJL ENTER LANGUAGE =
PCL&#13;&#10;<ESC/>&amp;l1T<ESC/>*t600R<ESC/>&amp;u600D<ESC/>&amp;l1X
```

For more information about escape characters for XML, see the *XML Specification* at www.w3c.org

.

### Modifying sequences

XDC Editor lets you modify the PDL commands that are associated with predefined sequence names.

1) At the bottom of the XDC Editor panel, click the Sequences tab.
2) Select the sequence to modify.
3) In the Sequence panel, modify the PDL statements as required. Use escape sequences to replace characters that cannot appear in XML documents. For information about escape sequences, see EncodingPCL sequences. For information about PDL sequences, see the corresponding reference documentation.

4) Click Apply.

# 20.3. Deploying XDC files

After creating a new XDC file or modifying an existing XDC file, make the file available to the services and form authors that use them. XDC Editor lets you edit the XDC files on your local hard drive.

If you are using SAP NetWeaver 7.32, copy the new or modified XDC file from your local hard drive to the /*[DIR_GLOBAL]*/AdobeDocumentServices/lib folder. If you are using SAP NetWeaver 7.1, use the Adobe document services configuration pages in the Web Administrator to upload new or modified XDC files from your local hard drive. If you are creating a new XDC file, you must also update table TSP0B to show the mapping between the new XDC file and the SAP device type that uses it. For information about mapping to XDC files on the Adobe document services for SAP device types, see "Administering XDC Files for SAP Device Types (Report RSPO0022)", which is available at the SAP HelpPortal

.

After creating a new XDC file or modifying an existing XDC file, make the file available to the services and form authors that use them. XDC Editor lets you edit the XDC files in a AEM forms server application or on your local hard drive.

Before the modified file can be used for printing, deploy it to Output. If you modified the media sizes in an XDC file and you want those sizes to be selectable from the Designer user interface, you must also deploy the modified file to your Designer installations.

## Deploying XDC files to Designer installations

By default, Designer uses the XDC file called Designer.xdc. This file is located in the Designer installation folder. Initially, this file is a default setup file installed with Designer; however, form authors can use other XDC files when printing a form. Typically, form authors use the appropriate printer-specific XDC file when testing their form design to make sure they get the expected results with the target printer.

If a form author requires Designer to use the settings in a new or modified XDC file for testing purposes, copy the modified XDC file to this Designer installation folder. For Designer installed separately from Workbench, place the modified file here:

- *[Installation Directory]* \Designer

  *For Designer installed with Workbench, place the modified file here:*
- *[Installation Directory]*\Workbench\Designer\

  *When form authors print a form with data, they can select the updated XDC file from a drop-down list in the Print dialog box to verify the output of the form design and data on the target printer. For information about print form designs with data, see Designer Help*

  .

### Making XDC files available to Output

When you use XDC Editor to modify an XDC file, the resultant file is stored on the local file system. To allow Output to access your modified XDC files, you must check them in to the AEM forms server (preferred) or copy them to a location that is accessible from AEM forms server. Here are some examples:

- A computer that has a folder shared with the server computer. In this case, the process or API references the modified XDC file by using the literal expression:

  *FILE:///Dir_1/Dir_2/myFile.xdc*

  *HTTP://name/myFile.xdc*
- A folder on the computer that hosts AEM forms server.

**To copy an XDC file from your local file system to the AEM forms server:**

1) Right-click the XDC file, and select Check In. See Workbench Help
2) for more information about checking in resources into the AEM forms server.

## 20.4. Determining PCL font sequences

This content explains how to determine the font setup sequences for PCL printers. In most situations, XDC Editor can determine the font setup sequence for you. However, you may need to add typeface codes to a sequence or, in rare situations, determine your own sequence.

### About escape sequences that specify printer-resident fonts

Escape sequences specify characteristics about fonts, which are listed in the order in which they are conventionally specified in the escape sequence:

- Text-parsing method that specifies whether character codes are interpreted as 1-byte or 2-byte character codes. (See Text-parsingmethod for fonts.)
- Character encoding or symbol set of the font family, such as ISO-8859-1, Windows 3.0 (Latin 2), and Big5. (See Characterencoding.)
- Font spacing, which can be constant or proportional. (See Fontcharacteristics.)
- Font pitch (monospace fonts) or height (variable-width fonts). The value must be supplied as the `fontPitch` or `fontSize` variable, depending on the type of font. When the form is printed, these variables are replaced with the specific settings that are specified in the form itself. (See Fontcharacteristics.)

**fontPitch**

Specifies pitch or characters per inch. Use this variable for monospace fonts such as Courier and Letter Gothic. If you omit this variable from a typeface definition, the default printer value is used. If you hardcode this value in your escape sequence, only the printer-installed font with that characteristic is used.

**fontSize**

Specifies font height. Use this variable for variable-width fonts such as Times New Roman. If you omit this variable from a typeface definition, the default printer value is used. If you hardcode this value in your escape sequence, only the printer-installed font with that characteristic is used.

- Style, such as upright, italic, or condensed. If not specified, the upright style is used. (See Fontcharacteristics.)
- Stroke weight, such as normal and bold. If not specified, the normal stroke weight is used. (See Fontcharacteristics.)
- Font family, such as Courier and Coronet. (See Fontcharacteristics.)

  *Most printer manuals provide the escape sequences to use to specify fonts. (See Findingdocumentation for your printer.)*

## Text-parsing method for fonts

The following table provides examples of sequences that specify text-parsing methods for fonts.

| Character encoding | Escape sequence |
|---|---|
| 1 byte | `<ESC/>&t0P`<br>or<br>`<ESC/>&t1P` |
| 1 or 2 bytes | `<ESC/>&t21P` |

## Character encoding

The following table provides examples of escape sequences that specify character encoding or symbol set for fonts.

| Character encoding | Escape sequence |
|---|---|
| Windows 3.0 | `<Esc/>(19U` |
| ISO 8859-2 (Latin2) | `<Esc/>(2N` |
| Shift-JIS | `<Esc/>(19K` |
| GBK | `<Esc/>(18C` |
| Hanguel | `<Esc/>(19H` |
| Big5 | `<Esc/>(18T` |
| Symbol | `<Esc/>(19M` |

## Font characteristics

The following table shows the Lexmark 644 escape sequence for typeface characteristics. The sequences must be XML-compatible, as described in EncodingPCL sequences.

| Characteristic | Escape sequence, (# is a placeholder) | Example | Meaning |
|---|---|---|---|
| Font spacing, either fixed or proportional | `<ESC/>(s#P` | `<ESC/>(s1P` | Proportional spacing |
| Type face (font family) | `<ESC/>(s#T` | `<ESC/>(s4099T` | Courier typeface |
| Characters per inch, used with monospace fonts | `<ESC/>(s#H` | `<ESC/>(s<var name= "fontPitch"/>H` | The variable is replaced to reflect the font size in the form. |
| Character height, used with variable-width fonts | `<ESC/>(s#V` | `<ESC/>(s<var name = "fontSize"/>V` | The variable is replaced to reflect the font size in the form. |
| Font style | `<ESC/>(s#S` | `<ESC/>(s0S` | Upright style |
| Stroke weight | `<ESC/>(s#B` | `<ESC/>(s3B` | Bold |
| Type face (font family) | `<ESC/>(s#T` | `<ESC/>(s4099T` | Courier typeface |

## Combining escape sequences

You can combine PCL emulation commands by linking them if the first 3 bytes of the commands are identical. The combined short form sends the first 3 bytes only once in the string. To combine commands, do these tasks:

- Use the first 3 bytes (characters) of the command only once at the start of the command string.
- Make the last letter of each command in the string lowercase.
- Capitalize the last letter of the string.

  *With some PCL printers, you can combine multiple commands with the encoding setting. The following command is a concatenation of two other commands:*

  `<ESC/>(s<var name = "fontPitch"/>H4099T`

  *It is 3 bytes shorter than the long form:*

  `<ESC/>(s<var name = "fontPitch"/>H<ESC/>(s4099T`

### Combined sequence that specifies the Courier font

The following sequence specifies the Courier printer-resident font on a Lexmark 644 PCL printer:

```
<ESC/>&amp;t0P<ESC/>(19U<ESC/>(s0p<var name="fontPitch"/>h0s0b4099T
```

The following table describes the meaning of each sequence expression in the above example.

| Sequence expression | Meaning |
|---|---|
| `(<ESC/>&t0P)` | Character codes are interpreted as 1-byte characters (text-parsing method). |
| `(<ESC/>(19U)` | Windows 3.0 character encoding. |
| `(<ESC/>(s0p<var name="fontPitch"/>h0s0b4099T)` | A combined escape sequence that comprises the following parts (shown in their non-combined form):<br>• *`<ESC/>(s0P`* specifies constant primary spacing (monospace).<br>*`<ESC/>(s<var name="fontPitch"/>H` specifies the number of characters per inch. (See Useof variables in sequences.)*<br>*`<ESC/>(s0S` specifies the font style as upright (default).*<br>*`<ESC/>(s0B` specifies a medium stroke weight (default).*<br>*`<ESC/>(s4099T` specifies the Courier font.* |

### Combined sequence that specifies the Coronet font

The following sequence specifies the Coronet printer-resident font on a Lexmark 644 PCL printer:

```
<ESC/>&amp;t0P<ESC/>(19U<ESC/>(s1p<var name="fontSize" />v1s0b4116T
```

The following table describes the meaning of each sequence expression in the above example.

| Sequence | Meaning |
|---|---|
| `(<ESC/>&t0P)` | Character codes are interpreted as 1-byte characters (text-parsing method). |
| `(<ESC/>(19U)` | Windows 3.0 character encoding. |

| Sequence | Meaning |
|---|---|
| `<ESC/>(s1p<var name="fontSize" />v1s0b4116T` | A combined escape sequence that comprises the following parts (shown in their non-combined form):<br>• *`<ESC/>(s1P`* specifies proportional primary spacing.<br>*`<ESC/>(s<var name="fontSize"/>V`* *specifies font height in points. (See Useof variables in sequences.) The variable value is supplied by Adobe document services.*<br>*`<ESC/>(s<var name="fontSize"/>V`* *specifies font height in points. (See Useof variables in sequences.) The variable value is supplied by the Output server.*<br>*`<ESC/>(s1S`* *specifies font style as italic.*<br>*`<ESC/>(s0B`* *specifies medium (default) stroke weight.*<br>*`<ESC/>(s4116T`* *specifies Coronet font.* |

# 21. Variable types reference

Provides descriptions of all the predefined types of process variables that you can create using Workbench. This section also provides the format to use for expressing the literal values of the variables when possible, such as in XPath expressions.

**NOTE:** The properties of the `int`, `boolean`, `byte`, `long`, `double`, `decimal`, `short`, and `float` types are the same as the properties of the Java primitive data types of the same name.

## 21.1. Service-defined data types

Some services require data as input and provide output data that are represented by a data type that the service defines. For example, the Output service returns operation results as data of type OutputResult.

The variables described in this section are not available for all services.

## 21.2. Complex data types

Complex data types store more than one data item. The data items appear below the variable in the process data model.

The data items that a complex variable stores are accessible using XPath expressions in the following format:

```
/process_data/variable_name/object/@data_item
```

*variable_name* is the name of the variable, and *data_item* is the name of the data item.

**NOTE:** If the data item is a `list` or `map` variable, use a different expression for accessing the data. (See Accessingdata in data collections.)

RELATED LINKS:

Process data model

Creating XPath expressions

## 21.3. Common variable properties

Some properties are common to all variable types. These properties are described in this section instead of in each operation section in the reference.

### Implicit properties

All service operations have the implicit properties Name, Title, Description, and Type. The `list`and `map` variable types are collections and have the Subtype property, which defines the type of data that they contain.

### Name

A name that identifies the variable in the process data model. The name must be a valid XML element name and therefore cannot contain spaces or special characters. The name must be unique to the process.

### Title

(Optional) A name that identifies the variable if it is exposed in a user interface such as Workspace. If no value is provided for Title, the value of the Name property is used by default.

### Description

A brief explanation of the purpose or contents of the variable. The description is useful for communicating the purpose of the variable to other process developers.

### Type

The variable type, which coincides with the type of data that it stores. For example, a `string` variable stores string data.

### Subtype

(Collections) The type of data that `list` and `map` variables store.

## General

Properties that specify whether the variable is involved in storing data when the process is initiated or is completed.

### Required

Select this option to specify that the data this variable stores must be provided when the process is invoked. If this option is not selected, providing the data when invoking the process is optional.

### Input

Select this option to specify that the variable is a parameter that data is provided for when invoking the process. The variable stores the data when the process is invoked. For example, if the purpose of a process is to manipulate a PDF document, create a `document` variable and select the In property.

### Output

Select this option to specify that the data the variable stores is provided in the process results. The results are available to the client that invoked the process when the process is complete.

### Enduser UI items

Properties that specify whether Workspace users can interact with the data that variables store. These properties are applicable to these variable types:

- boolean
- byte
- date
- date-time
- decimal
- double
- float
- int
- long
- short
- string

#### Searchable

Select this option to make the variable available as search criteria for Workspace users.

#### Visible In UI

Select this option to display the variable as task data to users in Workspace.

## 21.4. AcrobatVersion

A `string` value that represents the minimum Acrobat and Adobe Reader version required to view a PDF form that the Forms service renders. In addition, the PDF version used to create the PDF form is specified. `AcrobatVersion` values are used to configure the Acrobat Version property in the renderPDFForm operation of the Forms service.

For information about data that can be accessed using Xpath Expressions, see Dataitems.

### Data items

`AcrobatVersion` data values store one of these string values:

**auto:**

Target Version setting in the form design determines minimum version of Acrobat or Adobe Reader. In addition, the form design determines the PDF Version that is used to generate the PDF document.

**Acrobat_6:**

**Acrobat_7_0:**

PDF Version 1.6 is used to generate the PDF document. Adobe Reader and Acrobat 7.0 is the minimum version required to open the PDF document.

**Acrobat_7_0_5:**

PDF Version 1.65 is used to generate the PDF document. Adobe Reader and Acrobat 7.0.5 is the minimum version required to open the PDF document.

**Acrobat_8:**

PDF Version 1.7 is used to generate the PDF document. Adobe Reader and Acrobat 8.0 is the minimum version required to open the PDF document.

**Acrobat_8_1:**

PDF Version 1.7-ADBE-1 is used to generate the PDF document. Adobe Reader and Acrobat 8.1 is the minimum version required to open the PDF document.

**Acrobat_9:**

PDF Version 1.7-ADBE-3 is used to generate the PDF document. Adobe Reader and Acrobat 9.0 is the minimum version required to open the PDF document.

**Acrobat_10:**

PDF Version 1.7-ADBE-3 is used to generate the PDF document. Adobe Reader and Acrobat X is the minimum version required to open the PDF document.

# 21.5. AcrobatVersion-OutputService

A `string` value that represents the minimum Acrobat and Adobe Reader version required to view a PDF document. `AcrobatVersion-OutputService` values are used to configure the Acrobat Version property in the generatePDFOutput operation of the Output service.

For information about data that can be accessed using XPath Expressions, see Dataitems

For information about configuring default properties, see Datatypespecific settings.

## Data items

`AcrobatVersion-OutputService` data values store one of these string values:

**auto:**

Target Version setting in the form design determines minimum version of Acrobat or Adobe Reader. In addition, the form design determines the PDF Version that is used to generate the PDF document.

**Acrobat_6:**

**Acrobat_7_0:**

PDF Version 1.6 is used to generate the PDF document. Adobe Reader and Acrobat 7 is the minimum version required to open the PDF document.

**Acrobat_7_0_5:**

PDF Version 1.65 is used to generate the PDF document. Adobe Reader and Acrobat 7.0.5 is the minimum version required to open the PDF document.

**Acrobat_8:**

PDF Version 1.7 is used to generate the PDF document. Adobe Reader and Acrobat 8 is the minimum version required to open the PDF document.

**Acrobat_8_1:**

PDF Version 1.7-ADBE-1 is used to generate the PDF document. Adobe Reader and Acrobat 8.1 is the minimum version required to open the PDF document.

**Acrobat_9:**

PDF Version 1.7-ADBE-3 is used to generate the PDF document. Adobe Reader and Acrobat 9 is the minimum version required to open the PDF document.

## Datatype specific settings

Property for specifying the minimum Acrobat and Adobe Reader version required to view the PDF document and PDF version used to generate the PDF document.

### Default value

Sets the minimum version required to view a PDF document and the PDF version used to generate the PDF document. Select one of these values:

**<Use Form Template Default>:**

The Target Version setting in the form design determines the minimum version of Acrobat or Adobe Reader. Also, the form design determines the PDF Version.

**Acrobat and Adobe Reader 6 and above:**

PDF Version 1.5 is used to generate the PDF document.

**Acrobat and Adobe Reader 7.0 and above:**

PDF Version 1.6 is used to generate the PDF document.

**Acrobat and Adobe Reader 7.0.5 and above:**

PDF Version 1.65 is used to generate the PDF document.

**Acrobat and Adobe Reader 8 and above:**

PDF Version 1.7 is used to generate the PDF document.

**Acrobat and Adobe Reader 8.1 and above:**

PDF Version 1.7-ADBE-1 is used to generate the PDF document.

**Acrobat and Adobe Reader 9 and above:**

PDF Version 1.7-ADBE-3 is used to generate the PDF document.

# 21.6. ApproverTO

A complex data type that represents an approver in the approval stage of a review. Used with the Review, Commenting, and Approval building block, which is installed with the Managed Review & Approval Solution Accelerator.

For information about data that can be accessed using XPath Expressions, see Dataitems.

## Data items

The data items that ApproverTO values contain.

### additionalMetadata

A `string` value that represents additional information associated with the approver.

### completedBy

A `string` value that represents the user who approved the review content.

### completedFromIP

A `string` value that represents the IP address of the client machine that the review content was approved from.

### createdAt

A `string` value that represents how the approver was added to the review. Valid values are:

**TEMPLATE_CREATION**

Approver was part of the review template.

**INITIATION**

Approver added when review was initiated.

**UPDATES**

Approver added using addApprover(update) API.

**disposition**

A `string` value that represents a custom status for the approver.

**endDate**

A `date` value that specifies the end date of the approval task.

**finalComments**

A `string` value that represents the approver's final comments, which are added when completing the approval task.

**id**

Internal use only.

**reviewContext**

A ReviewContextTO value that represents the instance of the review.

**reviewStage**

A ReviewStageTO value that represents a stage of a multi-staged review and approval process.

**startDate**

A `date` value that specifies the start date of the approval task.

**status**

A `string` value that represents the status of the approval. The following values are valid:

- APPROVED
- REJECTED
- PENDING
- TERMINATED
- REMOVED

**umOid**

A `string` value that represents the User Management unique identifier for the approver.

# 21.7. AssemblerOptionSpec

A complex data type that represents the environment options for the invokeDDX and the invokeDDXOneDocument operations of the Assembler service. The data items include the default font to use, the logging level, and flags that specify how the operation will behave when an error occurs.

For information about data that can be accessed using Xpath Expressions, see Dataitems.

For information about configuring default properties, see Datatypespecific settings.

## Data items

The data items that `AssemblerOptionSpec` variables contain.

### defaultStyle

A `string` value that represents the default font style to use when none is specified in the DDX. The font style is the same format as the font attribute for DDX rich text elements: `"font: <font-style> <font-weight> <font-size>/<line-height> <font-family-specifier>"`. If the `<font-family-specifier>` contains any spaces, it should be surrounded by single quotation marks. For example, use `'Minion Pro'`. If no default is specified, a default of `"font: normal normal 12pt 'Minion Pro'"` is used.

### failOnError

A `boolean` value that specifies whether the operation stops executing immediately when an error occurs. A value of `true` means that the job is canceled when an error occurs. A value of `False` means that when an error occurs, the operation continues to execute.

### firstBatesNumber

The first `numeric` value for the first Bates number pattern on the first page, if the start value is not specified. The default for all other Bates number patterns is 1.

### logLevel

A `string` value that specifies the logging level. The value can be either an integer or its equivalent string value. The default setting for the logging level is INFO.

| String value | Integer | Description |
|---|---|---|
| OFF | positive infinity | Turn off logging of messages on the server. |
| SEVERE | 1000 | Log only messages that indicate a serious failure. |
| WARNING | 900 | Log only messages that indicate a potential problem. |
| INFO | 800 | Log only informational messages. |
| CONFIG | 700 | Log only static configuration messages. |

| String value | Integer | Description |
|---|---|---|
| FINE | 500 | Log tracing information. |
| FINER | 400 | Log fairly detailed tracing information. |
| FINEST | 300 | Log highly detailed tracing information. |
| ALL | negative infinity | Log all information. |

**NOTE:** The FINE, FINER, and FINEST settings should be used only when investigating specific problems because of their negative effect on performance.

**validateOnly**

A `boolean` value that specifies whether the operation is a validation-only test. In a validation-only test, the Assembler service only validates the DDX file but does not execute it. A value of `true` means that the operation is a validation-only test. A value of `false` means that the DDX file will be executed.

## Datatype specific settings

Environment properties for the invokeDDX and invokeOneDocument operations.

**Job Log Level**

A `string` value that specifies the type of message logging. These are valid string values:

**OFF:**

Messages are not logged.

**SEVERE:**

Only the messages that indicate a serious failure are logged.

**WARNING:**

Only the messages that indicate a potential problem are logged.

**INFO:**

Only the informational messages are logged.

**CONFIG:**

Only the static configuration messages are logged.

**FINE:**

Tracing information is logged.

**FINER:**

Fairly detailed tracing information is logged.

**FINEST:**

Highly detailed tracing information is logged.

The default value is INFO.

**Validate Only**

Determines whether this is a validation-only operation. When selected, the operation will only test the DDX file for validity, not execute it.

**Fail On Error**

Determines whether the operation is cancelled when an error occurs. When selected, the operation stops executing when an error occurs.

This option is selected by default.

**Default Style**

The default font style to use when none is specified in the DDX file.

**First Bates Number**

The first `numeric` value for the first Bates number pattern on the first page, if the start value is not specified.

## 21.8. AssemblerResult

A complex data type that stores the results of the invokeDDX operation that the Assembler service provides. It contains the successful document, the documents described by the DDX file (including PDF document attachments, if specified), throwable exceptions that occurred during the process, and the audit list.

An audit list is provided as a `document` value that is accessed using the `Job Log` data item.

The setting of the `failOnError` data item (an input option passed using an `AssemblerOptionSpec` data type) determines what occurs if exceptions occur when an Assembler service operation runs.

- When the `failOnError` data item is set to `True` (default) and an exception occurs, the operation throws the exception. If the thrown exception is caught, it can be extracted from the Job Log.
- When the `failOnError` data item is set to `False` and an exception occurs, the exception is not thrown. Instead, the operation completes normally and the exception is accessible from the `throwables` data item and seen in the Job Log. If more than one result block exists and some result blocks succeed and some fail, you will see values in both the `throwables` and `documents` data items, and also the exceptions in the Job Log.

For more information about DDX elements, see the DDX Reference.

For information about data that can be accessed using Xpath Expressions, see Dataitems.

## Data items

The data items that `AssemblerResult` variables contain.

### documents

A `map` of `string` values that contains the successful documents described by the result blocks in the DDX file. The key is a logical name for the document and the value is the file name of the document.

The logical name depends on the results:

- If the result is a single document, the logical name is the value of the `result` attribute. For example, if the DDX file contained `<PDF result="doc1.pdf">`, the logical name would be doc1.pdf.
- If the result is multiple documents that come from either a `<PackageFiles>` or a `<FileAttachments>` result element, then the logical document name is the last portion of a name after the last slash (/) if a path is specified. For example, if the document path provided is /dirABC/ABC.pdf, specify the XPath expression *[documents]*`[@id='ABC.pdf']` as a key to retrieve the document `value`, where *[documents]* is the name of a `map` variable.
- If the result is multiple from disassembling with the `<PDFsFromBookmarks>` element, then the logical name is the `prefix` attribute.

**NOTE:** Logical names for attachments are created by the Assembler service. You can access the list of names from the `successfulDocumentNames` data item.

### failedBlockNames

A `list` of `string`values that contains the names of the result blocks that failed.

### jobLog

A `document` value that contains the job log that was generated during the execution of the DDX.

### lastBatesNumber

A `numeric` value that represents the numeric portion form the last page stamped with a Bates number.

### multipleResultsBlocks

A `map` of `string` values that contains the result block names. Each name is mapped to a list of document names produced by that result block.

### numRequestedBlocks

An `int` value that represents the number of result blocks in the DDX.

### successfulBlockNames

A `list` of `string` values that contains the names of the result blocks that were successful.

**successfulDocumentNames**

A `list` of `string` values that contains the names of the documents that were successful.

**NOTE:** The Assembler service creates unique names for attachments using the naming scheme of [file-name of source document]_attach.[page of attachment].[index of attachment on a page] where:

*[file name of source document]* is based on the following rules:

- The file name of the source document. If a URL was provided, the last segment after the last slash in a URL is used. For example, for the URL of `http://www.adobe.com/go/ABC.pdf`, the file name `ABC.pdf` is used.
- The document name from the last path segment of the `PDF source` attribute in the DDX file.

*[page of attachment]* is the page number on which the attachment appears in the source document. For example, if an attachment appears on page 7, the value would be `0007`.

*[index of attachment on page]* is the index number of the attachment. This value is important when more than one attachment appears on a page. For example, if two attachments appear on a page, the first one has a value of `0001` and the second one has a value of `0002`.

For example, if a source document named PDF123.pdf had two attachments on the first page and one attachment on the third page, the names of the attachments would be `PDF123.pdf_attach.0001.0001`, `PDF123.pdf_attach.0001.0002`, and `PDF123.pdf_attach.0003.0001`.

**throwables**

A `map` of `string` values that contains the exceptions that were generated during execution of the DDX. The key in the map is the result block name in the `failedBlockNames` attribute. This data item is available only when you set the `failOnError` data item to a value of `False`. The `failOnError` data item is an input option passed using an `AssemberOptionSpec`.

# 21.9. binary

Holds binary data. When AEM Forms Server encounters data stored in a `binary` variable, it coerces the data to byte arrays. Use this variable type when you do not want data to be coerced to document or object values.

# 21.10. boolean

Holds a `boolean` value of `true` or `false`.

For information about configuring default properties, see Datatypespecific settings.

## Datatype specific settings

Select Value True to set the default value to `true`. Deselect Value True to set the default value to `false`.

RELATED LINKS:

false

true

# 21.11. byte

A byte of binary data.

# 21.12. Business Calendar Date

A complex data type that represents a date in a business calendar. A business calendar date is calculated using the specified number of days, the start date, and a business calendar, which is configured on AEM Forms Server.

The `Business Calendar Date` data type can be used as the values of operation properties that accept a `dateTime` value. Business dates that are calculated from a `Business Calendar Date` value are automatically coerced to a dateTime value.

For information about data that can be accessed using Xpath Expressions, see Dataitems.

For information about configuring default properties, see Datatypespecific settings.

## Data items

The data items that `Business Calendar Date` variables contain.

### calendar

A `string` value that represents the name of the business calendar to use.

### date

A `dateTime` value that represents the calculated business calendar date. This date is calculated when the property is requested.

### days

A `long` value that represents the number of business days. This value is used in the calculation of the date property.

### endTime

A `dateTime` value that represents the time of day that is assigned to the date property for a `Business Calendar Date` value. Though this is a `dateTime` value, only the time portion of the value is used that includes the hours, minutes, and seconds.

### startDate

A `dateTime` value that represents the starting date for the business calendar calculation.

## Datatype specific settings

Properties for business days and business calendar to use.

### Number of Business Days

An `int` value that represents the number of business days.

### Business Calendar Name

A `string` value that represents the name of the business calendar to use. A business calendar is configured on AEM Forms Server.

# 21.13. CentralResult

A complex data type that represents the result of invoking Adobe Central Pro Output Server (Central). The result includes output file, log, response, and trace files. `CentralResult` variables are used as values for output properties for the following Central Migration Bridge service operations:

centralMergeoperation

centralTransformationoperation

centralXMLImportoperation

You use the `CentralResult` variable to access the result of invoking Central commands. You also use it to access the log, trace, and response files associated with the invoked Central command.

For information about data that can be accessed using Xpath Expressions, see Dataitems.

## Data items

The data items that `CentralResult` variables contain.

### logDoc

A `document` value that represents a log file that contains entries for all activities that occur when Central is invoked. The log file includes trace, informational, warning, error, and severe messages from the Central Print Agent.

For more information about Central log files, see “-all (Log File Name)” in Print Agent Reference.

**responseDoc**

A `document` value that represents the response file that records an exit status. The default name of the response file that Print Agent creates is jetform.rsp. The jetform.rsp file is created in the same location as the Print Agent executable.

For information about response files, see "-arx (Response File)" in Print Agent Reference.

**resultDoc**

A `document` value that represents the output of a CentralMigrationBridge operation. Depending on the CentralMigrationBridge operation that is used, the results are as follows:

**centralMerge:**

A file provided in a file format supported by Central, such as PDF, PCL, IPL, PS, or ZPL.

**centralTransformation:**

The transformed data file.

**centralXMLImport:**

A data file (*.dat) file.

**traceDoc**

A `document` value that represents the trace file. The trace facility is commonly used for troubleshooting. However, it can also be used to generate data files that can be used as preamble for subsequent processing in the centralMerge operation. A *preamble* contains a series of commands that are executed before a form template and data file are merged. These commands often contain `^define` commands that establish event handlers that process `^group` commands.

For information about trace files, see "-atf (Trace File Name)" in Print Agent Reference.

# 21.14. Certificate Encryption Option Spec

A complex data type that represents the PDF encryption options for specifying the earliest version of Acrobat compatibility and the PDF document resources to encrypt. `Certificate Encryption Option Spec` values can be used as input properties for the CertificateEncrypt PDF operation that the Encryption service provides.

For information about data that can be accessed using Xpath Expressions, see Dataitems.

## Data items

The data items that `Certificate Encryption Option Spec`variables contain.

**compat**

A `Certificate Encryption Compatibility`value that specifies the earliest version of Acrobat that is compatible with the encryption used. This option also provides finer grained control over which functionality are permitted with the PDF document These string values are valid:

**ACRO_6:**

Compatible with Acrobat 6 and later versions that encrypts PDF documents using a 128-bit RC4 algorithm.

**ACRO_7:**

Compatible with Acrobat 7 or later versions that encrypts PDF documents using 128-bit Advanced Encryption Standard (AES).

**ACRO_9:**

Compatible with Acrobat 9 or later versions that encrypts PDF documents using 256-bit Advanced Encryption Standard (AES).

**option**

A `Certificate Encryption Option` value that specifies the PDF document resources to encrypt. These string values are valid:

**ALL:**

The entire PDF document is encrypted.

**ALL_EXCEPT_METADATA:**

The entire PDF document except for the metadata is encrypted.

**ATTACHMENTS_ONLY:**

Only the PDF document attachments are encrypted.

# 21.15. CertificateInformation

A complex data type that provides information about a certificate. Included in the information is the certificate path information, revocation information, and whether the certificate is a trusted anchor in AEM Forms.

`CertificateInformation` variables are members of the `CertificatePath` data type.

For information about data that can be accessed using Xpath Expressions, see Dataitems.

## Data items

The data items that `CertificateInformation` variables contain.

**certificate**

A `byte` value that contains information about the certificate path.

**revocationInformation**

A `RevocationInformation` value that contains revocation information for the certificate.

**trusted**

A `boolean` value that specifies whether the certificate is a trusted anchor configured in Trust Store on AEM Forms Server.

# 21.16. CertificatePath

A complex data type that represents the certificate path and information about the path's validation status and failure reason. `CertificatePath` variables are members of `IdentityInformation` variables.

For information about data that can be accessed using Xpath Expressions, see Dataitems.

## Data items

The data items that `CertificatePath` variables contain.

**certificateInformation**

A `list` of `CertificateInformation` values that represents certificates used to verify the identity of the signer.

**failureReason**

A `PathFailureReason` value that represents PKI failure reasons. The following reasons are valid:

**Invalid Name Constraints:**

An error occurred because an invalid name constraint was used.

**Invalid Basic Constraints:**

An error occurred because an invalid basic constraint was used.

**Missing Basic Constraints:**

An error occurred because basic constraints are missing.

**Invalid Policy Constraint:**

An error occurred because invalid policy constraints exist.

**Invalid Policy Mappings:**

An error occurred because invalid policy mappings exist.

**Unsupported Critical Extension:**

An error occurred because unsupported critical extensions exist.

**Invalid Key Usage:**

An error occurred because an invalid key was used.

**Path Length Constraint Not Satisfied:**

An error occurred because the path length constraint was not valid.

**Certificate Signature Does Not Match:**

An error occurred because the certificate signature does not match.

**Certificate Has Expired:**

An error occurred because the certificate expired.

**Unknown Reason:**

An error occurred for an unknown reason.

**Path Not Trusted:**

An error occurred because the path is not trusted.

**No Failure:**

No error occurred.

**IdenTrust Compliance Failed for Signing Certificate:**

An error occurred because verification of the IdenTrust Compliance failed.

**Revocation Checking of IdentTrust Certificates requires OCSP Request Signing Using an IdenTrust Credential:**

An error occurred because IdenTrust OCSP compliance failed.

**status**

An `IdentityStatus` value that represents whether the signer of the certificate is trusted.

# 21.17. Character Set (barcoded forms)

A `string` value that represents character set encoding. This variable type is used to configure Decode operations that the barcoded forms service provides.

These values represent the types of encoding that are valid for `CharSet` variables:

- UTF_8
- UTF_16
- ISO_8859_1
- ISO_8859_2
- ISO_8859_7
- Shift_JIS
- KSC_5601
- Big_Five
- GB_2312
- Hex (raw barcode data that is hex encoded)

## 21.18. Character Set (File Utilities)

A `string` value that indicates the character set to be used by the ReadString operation of the File Utilities service to read a document from the file system on AEM Forms Server.

These values are valid:

- UTF_8
- UTF_16BE
- UTF_16LE
- ISO_8859_1
- ISO_8859_2
- ISO_8859_7
- Shift_JIS
- KSC_5601
- Big_Five
- GB_2312

## 21.19. ContentAccessPermision (deprecated)

A complex data type that represents permissions that can be set for a s node. Setting permissions controls the user ability to perform tasks on content. For example, a permission may prevent the user from deleting content.

For information about data that can be accessed using Xpath Expressions, see Dataitems.

***NOTE:*** *Adobe is migrating Content Services ES customers to the Content Repository built on the modern, modular CRX architecture, acquired during the Adobe acquisition of Day Software. The Content Repository is provided with AEM Forms Foundation and is available as of the AEM Forms release.*

## Data items

The data items that `ContentAccessPermission` variable contain.

### authority

A `string` value that represents the authority that is associated with this permission.

### authority Type

A `string` value that represents the authority type that is associated with this permission.

### isAllowed

A `boolean` value that specifies whether the specified users or groups have the selected usage permissions for the folder or content. A value of true indicates that all the specified users or groups have the permission to use the folder or content. A value of false indicates that all the specified users or groups do not have the permission to use the folder or content.

### permission

A `string` value that represents the permission that is associated with this object. The permission can be one of the following values: Coordinator, Collaborator, Contributor, Editor, or Consumer.

RELATED LINKS:

Document Management

# 21.20. CRCResult (deprecated)

A complex data type that stores the results of the retrieveContent, storeContent, and getSpaceContents operations that the Document Management service (deprecated) provides. In the retrieveContent operation, the data type contains a document that is retrieved during a query operation. In the storeContent operation, the data type contains the properties of the created node. In the getSpaceContents, the data type contains a list of `CRCResult`, one for each document.

For information about data that can be accessed using Xpath Expressions, see Dataitems.

## Data items

The data items that `CRCResult` variables contain.

### attributeMap

A `map` of `string` values that contains the names and values of the attributes for the content. The content has the following attributes:

**creator:**

The creator of the content.

**browse-link:**

The link to where the content is stored.

**store-protocol:**

The protocol of the store that is used for the content.

**title:**

The title that is displayed in the web client for the content.

**node-uuid:**

A unique identifier value for the content.

**node-type:**

The content type.

**folder-path:**

The content path, up to the parent folder.

**store-identifier:**

A unique identifier value for the store.

**description:**

The description for the content.

**resolved-path:**

The fully resolved path with namespaces.

**browseLink**

A `string` value that specifies the browse link of the content. A browse link is a URL that you can enter into a web browser to display the content.

**contentMimeType**

A `string` value that specifies the MIME type of the content.

**creationDate**

A `dateTime` value that represents the date on which the content was created.

**creator**

A `string` value that specifies the creator of the content.

**document**

A `document` value that represents the content retrieved from Content Services.

**folderpath**

A `string` value that specifies the folder path of the content.

**modificationDate**

A `dateTime` value that represents the date on which the content was last modified.

**modifier**

A `string` value that represents the name of the user who last modified the content.

**nodeName**

A `string` value that represents the name of the content.

**nodeType**

A `string` value that represents the node type. It can be either content or folder.

**nodeUuid**

A `string` value that represents the identifier value of the content.

**qualifiedNodePath**

A `string` value that represents the fully qualified path of the content.

**storeName**

A `string` value that represents the name of the store in which the content is located.

**storeScheme**

A `string` value that represents the scheme of the store in which content is present.

**title**

A `string` value that represents the title of the content.

**versionLabel**

A `string` value that represents the version label of the content.

## 21.21. CreateDocumentResultType

A complex data type that represents the output of the Create Document operation in a SharePoint repository. This data type is used in the Create Document operation provided by the Connector for Microsoft SharePoint service.

### Data items

The data items that Create Document Result Type contains.

#### absoluteFileUrl

A `string` value that represents absolute URL of the uploaded file.

#### relativeFileUrl

A `string` value that represents the relative URL of the uploaded file.

#### fileUploadSuccessful

A `boolean` value that indicates whether the creation of the document was successful. A value of `true` indicates the document was created successfully, and `false` indicates that the document was not created.

## 21.22. Credential

A complex data type that represents a credential that is required to sign or certify a PDF document. This data type is used in the SignSignature Field operation provided by the Signature service.

For information about data that can be accessed using Xpath Expressions, see Dataitems.

For information about configuring default properties, see Datatypespecific settings.

### Data items

The data items that `Credential` variables contain.

#### alias

A `field name` value that represents the name used to identify the credential.

#### certificate

A `byte` value that represents the certificate required for signing when using the signing service provider interface (SPI).

#### credentialType

An `int` value that represents the credential type. These values are valid:

**ALIAS_CERTNKEY_NONE:**

The alias is provided by configuring the certificate and key values in Trust Store. The signing SPI is not queried for an alias.

**CERT_NONE_KEY:**

The certificate is not provided and Trust Store is not queried. The key value comes from the signing SPI.

**spiName**

A `string` value that represents the name of the signing SPI.

## Datatype specific settings

Properties for configuring credential details.

**Use SPI**

Select this option to use the credentials from the SPI. When this option is deselected, local credentials are used. By default, the option is deselected.

**Alias**

Sets an alternative name for a credential managed by the Signature service.

**SPI Name**

Sets the name of the signing server provider interface. This name is provided to the Signature service to extend the digital signature functionality when credentials are not exposed to AEM forms Server.

**Certificate**

Sets the location of the certificate on the file system. This option is available when the Use SPI option is selected.

When you click the ellipsis button , the Open dialog box opens. In the dialog box, you can select a file from your computer or from a network location. During run time, if you selected a file from a location on your computer, it must exist in the same location on AEM Forms Server.

**SPI Properties**

Sets the location of the properties file to pass custom inputs to an implementation of the SPI. This option is available when the Use SPI option is selected.

If you provide a literal value, clicking the ellipsis button opens the SPI properties dialog box. (See AboutSPI Properties.) In the dialog box, add, remove, and edit the keys and values for each SPI property. The SPI implementation determines the keys that you provide. For information about creating custom service providers, see Programming with AEM Forms

.

# 21.23. CRLOptionSpec

A complex data type that stores preferences for the certificate revocation list (CRL).

`CRLOptionSpec` variables are used in the following operations of the Signature service:

CertifyPDF

SignSignature Field

VerifyPDF Signature

VerifyPDF Signature (deprecated)

VerifyXML Signature

For information about data that can be accessed using Xpath Expressions, see Dataitems.

For information about configuring default properties, see Datatypespecific settings.

## Data items

The data items that `CRLOptionSpec` variables contain.

### alwaysConsultLocalURL

A `boolean` value specifies whether to use the CRL location provided as a local URI before any specified locations within a certificate. The CRL location provided is used for revocation checking. When this value is set to `true`, it means the local URI is used first. The default value of `false` indicates that locations specified in the certificate are used before the local URI is used.

### goOnline

A `boolean` value that indicates whether to access the network to retrieve CRL information. Accessing the network to retrieve the most recent CRL list can improve network performance by going online only when necessary. When the value is set to `false`, CRL information is not retrieved online. The default value of `true` indicates that CRL information is accessed online.

### ignoreValidityDates

A `boolean` value that indicates whether to use thisUpdate and nextUpdate times. Ignoring the response's thisUpdate and nextUpdate times prevents any negative effect on response validity. The thisUpdate and nextUpdate times are retrieved from external sources by using HTTP or LDAP and can be different for each revocation information. A value of `true` indicates that the validity dates are ignored. The default value of `false` indicates that validity dates are used.

### LDAPServer

A `string` value that represents the URL or path of the Lightweight Directory Access Protocol (LDAP) server. The LDAP server is used to retrieve information about the certificate revocation list (CRL). For example, you can type `www.ldap.com` for the URL or `ldap://ssl.ldap.com:200` for the path and

port. The LDAP server searches for CRL information using the distinguished name (DN) according to the rules specified in RFC3280, section 4.2.1.14.

### localURI

A `string` value that represents the URL for the local CRL store. This value is used only if the alwaysConsultLocalURL value is set to `true`. The default value is `null`.

### requireAKI

A `boolean` value that specifies whether an AKI extension must be present in a CRL. An *authority key identifier (AKI)* helps to identify the next certificate within a certificate chain. A value of `true` indicates that the AKI extension is required. The default value of `false` indicates that the AKI extension is not required.

### revocationCheckStyle

A `RevocationCheckStyle` value that specifies the type of revocation check that is performed when verifying a signature in a PDF document.

## Datatype specific settings

Properties for configuring the certificate revocation options.

### Consult Local URI First

Select this option to use the CRL location provided as a local URI before any specified locations within a certificate. The CRL location provided is used for revocation checking. When this option is selected, it means the local URI is used first. When this option is deselected, the locations specified in the certificate are used before the local URI is used. By default, the option is deselected.

### Local URI for CRL Lookup

Sets the URL for the local CRL store. This value is used only if the Consult Local URI First option is selected.

### Revocation Check Style

Sets the revocation-checking style used for verifying the trust status of the CRL provider's certificate from its observed revocation status. The default is BestEffort. Select one of these values:

**NoCheck:**

Does not check for revocation.

**BestEffort:**

Checks for revocation of all certificates when possible.

**CheckIfAvailable:**

Checks for revocation of all certificates only when revocation information is available.

**AlwaysCheck:**

Checks for revocation of all certificates.

**LDAP Server**

Sets the URL or path of the Lightweight Directory Access Protocol (LDAP) server used to retrieve information about the certificate revocation list (CRL). The LDAP server searches for CRL information using the distinguished name (DN) according to the rules specified in RFC3280, section 4.2.1.14. For example, you can type `www.ldap.com` for the URL or `ldap://ssl.ldap.com:200` for the path and port.

**Go Online for CRL Retrieval**

Select this option to access the network to retrieve CRL information. Accessing the network to retrieve the most recent CRL list can improve network performance by going online only when necessary. When this option is deselected, CRL information is not retrieved online. By default, the option is selected.

**Ignore Validity Dates**

Select this option to use thisUpdate and nextUpdate times. Ignoring the response's thisUpdate and nextUpdate times prevents any negative effect on response validity. The thisUpdate and nextUpdate times are retrieved from external sources by using HTTP or LDAP and can be different for each revocation information. When the option is deselected, the thisUpdate and nextUpdate time are ignored. By default, the option deselected.

**Require AKI Extension in CRL**

Select this option to specify that the Authority Key Identifier (AKI) extension must be present in the CRL. The AKI extension can be used for CRL validation. When this option is deselected, the presence of the AKI extension the CRL is not required. By default, the option is deselected.

# 21.24. CustomAttributeTO

A complex data type that represents custom attributes associated with a review or review template. Used with the Review, Commenting, and Approval building block, which is installed with the Managed Review & Approval Solution Accelerator.

For information about data that can be accessed using XPath Expressions, see Dataitems.

## Data items

The data items that CustomAttributeTO values contain.

**attrKey**

A `string` value that represents the custom attribute key.

### attrValue

A `string` value that represents the value of the custom attribute.

### id

Internal use only.

### reviewContext

A ReviewContextTO value that represents the version of the review that the custom attributes are associated with.

### reviewTemplate

A ReviewTemplateTO value that represents the review template that the custom attributes are associated with.

## 21.25. DataAndMetaDataDescriptor

A data type that stores the description of data and metadata retrieved for a PLM (Product Lifecycle Management) object.

## 21.26. date

Holds a date value.

### literal value

```
"yyyy-mm-dd"
```

where

- `yyyy` is the year.
- `mm` is the number that represents the month.
- `dd` is the calendar day of the month.

### Example

```
"2006-05-24"
```

RELATED LINKS:

Date and time parameters

# 21.27. dateTime

Holds a value that represents the date and time.

## Literal value

```
"yyyy-mm-ddThh:mm:ssZ"
```

where

- `yyyy` is the year.
- `mm` is the number that represents the month.
- `dd` is the calendar day of the month.
- `hh` is the hour, using the 24-hour format.
- `mm` is the minutes.
- `ss` is the seconds.
- `Z` is the time zone. See the `time zone` parameter in this table for more information.

## Example

```
"2006-03-02T14:37:00Z"
```

RELATED LINKS:

Date and time parameters

# 21.28. decimal

Holds a decimal value.

In the AEM Forms database, values are represented by an unscaled integer value and a scale that determines the location of the decimal. The `scale` attribute indicates the number of digits in the integer that are on the right side of the decimal. The default scale is `3`.

For example, an integer value of `12345` and a scale of `3` represents the decimal value of `12.345`.

The `length` attribute determines the maximum number of digits that can be stored. The default length is `10`.

You can change the values of the `scale` and `length` attributes using the Variables view.

## Literal value

A decimal number

### Example

```
123.4
```

## 21.29. Delimiter

A `string` value that represents the type of line delimiter used in the barcode data. It is used by the Extractto XML operation of the barcoded forms service when it converts delimited barcoded data into XML data. The default value is `Tab`.

These values are valid:

- Carriage_Return
- Colon
- Comma
- Line_Feed
- Pipe
- Semi_Colon
- Space
- Tab

## 21.30. document

Binary data that represents the deserialized version of a file. This variable is typically used to pass data of arbitrary size between service operations, such as PDF documents or XML documents.

Use document values to set the default value of an operation property that requires a document value. Create a document variable and set the Default Value property of the variable. Then, set the value of the operation property as the document variable.

Depending on the size of the data that a `document` variable represents, the data is either stored in memory or stored as a file in the file system of AEM Forms Server.

### Datatype Specific Settings

Set the value of the Default Value property to specify a file that is initially stored in the variable. Click the ellipsis button to browse for an asset in an application.

**NOTE:** A document input variable can reference a remote asset located in another application. If a user does not have sufficient rights to access the remote asset, it generates an error when it is used in a process. To avoid this problem, assign the required rights to the user. Alternatively, configure the variable as a type other than input.

# 21.31. DocInfo

A complex data type that stores the attributes and content for an item in the IBM Content Manager repository.

For information about data that can be accessed using Xpath Expressions, see Dataitems.

## Data items

The data items that `DocInfo` variables contain.

### attributeNameValue

A `map` value for mapping attributes of an item type to process variable values. The process variables that you map to attributes must be the same data type. For example, if an attribute is of data type `string`, it must be mapped to a process variable of data type `string`. The name of an attribute is used as the key.

### content

A `document` value that represents the contents for the item. For a folder, this value is empty.

### objectType

A `string` value that represents the item type of the item.

### pid

A `string` value that represents the unique identifier of an item. The PID (Persistent Identifier) is an identifier used by Content Manager server to uniquely identify all items.

### version

A `string` value that represents the version of the item in a Content Manager repository.

# 21.32. Document Form

**NOTE:** This variable type was deprecated in LiveCycle ES2.

A complex data type that holds form data (in binary format) and the URL of a PDF or XML form. The form can be merged with the data when the form is rendered at run time.

`Document Form` variables are typically used in human-centric processes where a PDF document must be created from the data and the form, and saved unaltered. In this case, the form is usually a PDF file. For example, a process requires that a user digitally signs a PDF form. The form is saved as a PDF document that is not altered thereafter.

By default, the render service that is used for `Document Form` variables is the invoke operation that the Default Render service provides. This service is a process that forms workflow provides. The process merges the variable's form data with the associated form.

No default submit service is configured.

For information about how to configure the General and Enduser UI Items properties, see Commonvariable properties.

For information about data that can be accessed using Xpath Expressions, see Dataitems.

For information about configuring default properties, see Datatypespecific settings.

For information about configuring properties for render and submit services, see AdvancedSettings.

## Data items

The data items that `Document Form` variables contain.

### document

A `document` value that represents form data that can be merged with the PDF form that the formURL references.

If the Call The Render Service Only Once option is selected, this value is the PDF document that is created from the PDF form and the form data.

### formURL

A `string` value that represents the URL of the PDF form.

Forms that are used with the Document Form data type must be designed to submit the PDF form rather than the XDP or XML data. (See "Preparing form designs for forms workflow" in Designer Help.)

Clients that use Adobe Reader must have the form rights-enabled to submit a PDF file. (See ReaderExtensions.)

## Datatype specific settings

Properties for specifying the form.

### URL

The URL of the PDF form that resides in the repository. This value is stored in the `formURL` data item.

## Advanced Settings

Properties for configuring the render and submit services.

## Data items

Properties for configuring the render service, which processes the form data before it is sent to Workspace.

### Enable Render Service

Determines whether the form data is rendered upon task creation. When selected, a render service is used. When not selected, no processing of the data is performed before it sent to the client.

This option is selected by default.

### Call The Render Service Only Once

Determines whether a PDF document is created from the data and the form. When selected, the render service is called only once to prevent the PDF document from being altered.

This option is selected by default.

### Service Name and Operation Name

The service and operation to use to render the form and form data that the variable holds.

The default service is the Default Render service, and the default operation is the invoke operation.

### Service Input

The input properties of the operation that is selected as the render service. The properties that appear depend on the render service that is selected.

The default value of each of the properties is determined by the Assign Task operation from the User service that uses the `Document Form` variable as its input form variable.

### Service Output

The output properties of the operation that is selected as the render service. The properties that appear depend on the render service that is selected.

For information about the input and output properties for the specified render service, see Servicereference. For information about the Default Render ES Update1 service, see DefaultRender ES Update 1.

### Submit Service

Properties for configuring the submit service, which processes the form data after it is submitted by the end user.

*Enable Submit Service*

Determines whether the form data is processed after the form is submitted. When selected, a submit service is used. When not selected, no processing of the data is performed after it is submitted.

This option is not selected by default.

*Service Name and Operation Name*

The service and operation to use to process the form data after it is submitted. There is no default submit service.

*Service Input*

The input properties of the operation that is selected as the submit service. The properties that appear depend on the submit service that is selected.

*Service Output*

The output properties of the operation that is selected as the submit service. The properties that appear depend on the submit service that is selected.

For information about the input and output properties for the specified submit service, see Servicereference.

# 21.33. DocumentTO

Internal use only.

# 21.34. double

Holds a double-precision floating point value (64-bit IEEE 754).

## Literal value

Any number between the values 4.94065645841246544e-324d and 1.79769313486231570e+308d (positive or negative).

**NOTE:** You cannot express double values using scientific notation, which XPath does not support.

# 21.35. Encryption Identity

A complex data type that represents an encryption identity, which describes a specific certificate and its permissions to encrypt a PDF document with.

For information about data that can be accessed using Xpath Expressions, see Dataitems.

## Data items

### perms

A `CertificateEncryptionPermission` value that stores the associated permissions for the certificate. A Certificate Encryption Permission represents the permissions that can be used to secure a PDF document. These string values are valid:

**PKI_ALL_PERM:**

All the permissions for a user are available. No restrictions have been placed on the document.

**PKI_EDIT_ADD:**

The permission for a user to add form data.

**PKI_EDIT_ASSEMBLE:**

The permission for a user to assemble PDF documents.

**PKI_EDIT_COPY:**

The permission for a user to copy form data.

**PKI_EDIT_EXTRACT:**

The permission for a user to extract data from forms.

**PKI_EDIT_FILL:**

The permission for a user to fill in a form.

**PKI_EDIT_MODIFY:**

The permission for a user to edit the PDF document.

**PKI_EDIT_HIGH:**

The permission to print using high resolution.

**PKI_EDIT_LOW:**

The permission to print using low resolution.

### recipient

A `Recipient` value, which is a complex data type that you use to store information used for encrypting a PDF document. `Recipient` values consist of the following data items:

**alias**

The `string` value that specifies the alias of the x509 certificate.

**ldapURI**

A `string` value that specifies URI of an LDAP server that determines the permissions for the certificate.

**x509Cert**

A `string` value that specifies the location of an x509 certificate.

# 21.36. EncryptionTypeResult

A complex data type that is used to store the results of the getPDFEncryption operation of the Encryption service.

For information about data that can be accessed using Xpath Expressions, see Dataitems.

## Data items

The data items that `EncryptionTypeResult` variables contain.

**encryptionType**

A `string` value that contains one of these values:

**PASSWORD:**

The PDF document is encrypted with a password.

**POLICY_SERVER:**

The PDF document is protected with a policy created by the Rights Management service.

**CERTIFICATE:**

The PDF document is protected with a certificate-based encryption.

**OTHER:**

The PDF document is protected with another type of encryption.

**NONE:**

The PDF document is not encrypted.

# 21.37. FieldMDPOptionSpec

A complex data type that represents the PDF document fields that are locked after an associated signature field is signed. Any changes to the locked fields after the signature field is signed invalidate the signature.

This data type is used in the AddInvisible Signature Field operation provided by the Signature service.

For information about data that can be accessed using Xpath Expressions, see Dataitems.

For information about configuring default properties, see Datatypespecific settings.

## Data items

The data items that `FieldMDPOptionSpec` variables contain.

### action

A `string` value that represents the set of fields that are locked when the signature field is signed. These string values are valid:

**ALL:**

All fields in the PDF document are locked.

**INCLUDE:**

Lock only the fields specified in the fields data item.

**EXCLUDE:**

Lock all fields except for those fields specified in the fields data item.

### fields

A `list` of `string` values that specifies the names of the fields that the specified action applies to. Each value is separated by a comma. This property is not used if the `action` property is set to ALL.

## Datatype specific settings

Properties for configuring the locked fields after signing the PDF document.

### Field Locking Action

A list that sets the type of action to use to lock fields in a PDF document. No default value is selected. Select one of these values:

**All Fields:**

Lock all fields in the PDF document.

**Include Fields:**

Lock only the fields specified in the Application To Form Fields option.

**Exclude Fields:**

Lock all fields except for those fields specified in the Applicable To Form Fields option.

### Applicable to Form Fields

Sets a comma-separated list of field names that indicate which fields the action is applicable or not applicable to. This option is available when Field Locking Action option is set to a value of Include Fields or Exclude Fields.

# 21.38. float

Holds a single-precision floating point value (32-bit IEEE 754).

## Literal value

Any number between the values 1.40129846432481707e-45 and 3.40282346638528860e+38 (positive or negative)

**NOTE:** You cannot express float values using scientific notation, which XPath does not support.

# 21.39. Form

**NOTE:** This variable type was deprecated in LiveCycle ES2.

A complex data type that you can use to store XML data and the location of a file in the repository.

`Form` variables are typically used in human-centric processes where a Flex application (SWF file) is used as a form. The variable stores the location of the Adobe Flex application (SWF file) and the XML data that is captured with the application.

By default, the render service that is used for `Form` variables is the Read Resource Content operation that the Repository service provides. The service retrieves the Adobe Flex application from the repository.

No default submit service is configured.

For information about how to configure the General and Enduser UI Items properties, see Commonvariable properties.

For information about data that can be accessed using Xpath Expressions, see Dataitems.

For information about configuring default properties, see Datatypespecific settings.

For information about configuring properties for the render and submit services, see Advancedsettings.

For information about configuring schema settings, see Schemasettings.

## Data items

The data items that `Form` variables contain.

### data

XML data that is associated with the form. The data items that are below this node depend on the data schema of the Adobe Flex application.

### formUrl

A `string` value that represents the URL of an Adobe Flex application (SWF file) that resides in the repository. The value is set using the URL property of this variable. However, you could set the value using XPath expressions at run time.

## Datatype specific settings

Properties for specifying the form and initial form data.

### URL

The URL of the Adobe Flex application that is associated with the form data that this `Form` variable holds. The Adobe Flex application must reside in the repository. This value is stored in the `formUrl` data item.

### Default Data

(Optional) Form data to use to populate the Adobe Flex application if no other data is stored in the variable. This data is stored in the `data` data item.

## Advanced settings

Properties for configuring the render and submit services.

### Render Service

Properties for configuring the render service, which processes the form data before it is sent to Workspace.

*Enable Render Service*

Determines whether the form data is rendered upon task creation. When selected, a render service is used. When not selected, no processing of the data is performed before it sent to the client.

This option is selected by default.

*Service Name and Operation Name*

The service and operation to use to render the form and form data that the variable holds.

The default service is RepositoryService, and the default operation is readResourceContent.

*Service Input*

The input properties of the operation that is selected as the render service. The properties that appear depend on the render service that is selected.

The default value of each of the properties is determined by the Assign Task operation from the User service that uses the `Form` variable as its input form.

*Service Output*

The output properties of the operation that is selected as the render service. The properties that appear depend on the render service that is selected.

For information about the input and output properties for the specified render service, see Servicereference. For information about the default render service, see ReadResource Content.

**Submit Service**

Properties for configuring the submit service, which processes the form data after it is submitted by the end user.

*Enable Submit Service*

Determines whether the form data is processed after the form is submitted. When selected, a submit service is used. When not selected, no processing of the data is performed after it is submitted.

This option is not selected by default.

*Service Name and Operation Name*

The service and operation to use to process the form data after it is submitted. There is no default submit service.

*Service Input*

The input properties of the operation that is selected as the submit service. The properties that appear depend on the submit service that is selected.

*Service Output*

The output properties of the operation that is selected as the submit service. The properties that appear depend on the submit service that is selected.

For information about the input and output properties for the specified submit service, see Servicereference.

## Schema settings

The XML schema that the Adobe Flex application data conforms to. The schema enables XPath Builder to represent the data in the process data model at run time. Seeing the form data in the model is useful for creating XPath expressions using XPath Builder.

Click Schema Settings to view the schema or select a schema file from the local file system.

RELATED LINKS:

Configuring XML variables for XDP data

# 21.40. FormDatatype

A `string` value that specifies the form type for storing form information. It is used by the Assign Task operation in the User service. The following values are valid: Form, xfaForm, and Document Form.

End Hidden section

# 21.41. FormModel

A `string` value that represents the location where the Forms service processes an HTML form. `FormModel` variables are used to configure the Form Model property of the (Deprecated)renderHTML-Form operation of the Forms service. In addition, `FormModel` variables are members of HTMLRender-Spec data types.

For information about data that can be accessed using Xpath Expressions, see Dataitems.

## Data items

`FormModel` data values store one of these string values:

**auto:**

The Forms service checks the form design to determine whether to render on the client or server.

**client:**

The form is rendered on the client. Scripts in a form that are run on both the server and the client generate a warning on AEM Forms Server.

**server:**

The form is rendered on the server.

**both:**

The form is rendered on both the server and the client.

# 21.42. FormPreference

A `string` value that represents the transformation types that are used by the RenderFormDocument, and (Deprecated) renderHTMLForm operations of the Forms service. A FormPreference variable can be used as the value of the Render As property of these operations.

Specifying a variable as the property value is recommended only if the variable value is passed into the process, for example as a process invocation parameter, or if you use the same value for the Render As property of several operations on your process diagram. For other situations, specifying the literal value of the property is easier to accomplish.

The following values are valid for the FormPreference variable:

**AHTML:**

An HTML transformation type that is compatible with accessibility-enhanced browsers (currently Internet Explorer 5 or later).

**AUTO:**

An HTML transformation type where the Forms service determines the best transformation.

**Guide:**

A PDF transformation type that uses a capability based on Flash technology for guiding a user to complete a data capture transaction.

**HTML4:**

An HTML transformation type that is compatible with older browsers that do not support absolute positioning of HTML elements.

**IMAGEREF:**

For internal use only.

**MSDHTML:**

An HTML transformation type that is compatible with dynamic HTML for Internet Explorer 5.0 or later.

**NoScriptXHTML:**

An HTML transformation type that is compatible with the CSS2 specification and compliant with XHTML 1.0.

**PDFForm:**

A PDF transformation type that renders interactive PDF form compatible with Acrobat Professional and Acrobat Standard, version 6.0.2 or later.

**PDFFormCC:**

For internal use only.

**PDFMerge:**

A PDF transformation type that merges data into a prerendered PDF form.

**StaticHTML:**

An HTML transformation type that does not allow users to enter data into fields on the HTML form.

**XDPPDFForm:**

For internal use only.

**XHTML:**

An HTML transformation type that is compatible with accessibility-enhanced browsers (currently Internet Explorer 5 or later).

**NOTE:** The (Deprecated) renderHTMLForm operation supports only the AHTML, AUTO, HTML4, MSDHTML, NoScriptHTML, StaticHTML, and XHTML transformations.

# 21.43. (Deprecated) FormGuideRenderSpec

A complex data type that stores run-time options that specify how the Forms service renders form guides. `FormGuideRenderSpec` variables are used to the configure Form Guide Render Options property of the renderFormGuide operation in the Forms service.

In AEM Forms version 8 and earlier, this data type was called ActivityGuideRenderSpec.

This data type is useful only when you are using the renderFormGuide operation with a form design and embedded guide created in Designer 8.2. Form designs created using Designer cannot be used with the renderFormGuide operation.

For information about data that can be accessed using Xpath Expressions, see Dataitems.

For information about configuring default properties, see Datatypespecific settings.

## Data items

The data items that `FormGuideRenderSpec` variables contain.

### CB

A `boolean` value that determines whether the Forms service generates a form guide with an HTML template. An HTML template is a HTML code that launches the form guide. A value of `true` indicates to generate an HTML template with the form guide and `false` indicates not to generate the HTML template with the form guide.

### guideAccessible

A `boolean` value that indicates whether the form guide is accessible. A value of `true` indicates the form guide is created to be accessible and `false` indicates that it is not.

### guideCBURL

A `string` value that represents the URL for a custom override implementation of the callback servlet.

### guideName

A `string` value that represents the name of the guide within the generated form guide.

### guidePDF

A `boolean` value that determines whether the output is rendered in the form guide and PDF form. A value of `true` indicates that both the form guide and PDF form are rendered and `false` indicates that only the form guide is rendered.

### guideRSL

A `boolean` value that determines whether to use run-time shared libraries when compiling a form guide. A value of `true` indicates that shared libraries are used and `false` indicates they are not used.

### guideStyle

A `string` value that represents the name of the style sheet that the form guide uses. The style sheet is also embedded in the form guide. The application container contains the style sheet, or the style sheet can be referenced.

### guideSubmitAll

A `boolean` value that determines whether the form guide submits all data including hidden panels. A value of `true` indicates that all data is submitted and `false` indicates that all data is submitted except for data from hidden panels.

### ifModifiedSince

For internal use only.

### injectFormBridge

A `boolean` value that determines whether the Forms service inserts special JavaScript™ code into a PDF form. The JavaScript code performs operations on a PDF form when the form guide is rendered. A value of `true` indicates that the Forms service inserts the JavaScript code and `false` indicates that it does not.

### locale

A `string` value that represents the locale used by the Forms service for sending validation messages to client devices. For a complete list of supported locale codes, see http://java.sun.com/j2se/1.5.0/docs/guide/intl/locale.doc.html.

## Datatype specific settings

Properties for configuring the form guide that the variable represents.

### If Modified Since

For internal use only.

### Guide Name

Sets the name of the guide from the form design. You can create more than one guide for each form design. The first guide is used if no value is provided.

### Guide RSL

Sets whether to use shared run-time libraries for compiling a form guide. The default value is False. Select one of these values:

**False:**

Do not use the shared run-time libraries.

**True:**

Use the shared run-time libraries.

### Guide PDF

Sets whether the PDF form is rendered with the form guide. The default value is True. Select one of these values:

**True:**

Render a PDF form with the form guide.

**False:**

Do not render a PDF form with the form guide.

### Guide Accessible

Sets whether the form guide compilation is accessible. The default value is False. Select one of these values:

**False:**

The form guide compilation is not accessible.

**True:**

The form guide compilation is accessible.

### Guide CB Url

Sets the location for a custom override implementation of the callback servlet. The value must be a valid URL expressed as a string value.

### Locale

Sets the language used to send validation messages to client applications, such as web browsers, when a form guide is rendered. The default value is <Use Server Default>. Select a language from the list or one of these values.

**<Use Server Default>:**

Use the Locale setting configured on AEM Forms Server. The Locale setting is configured in administration console. (See AEM Forms administration help .)

**<Use Custom Value>:**

Use a locale that is not available in the list. After selecting this option, in the text box beside the list, type the Locale ID of the locale code to use. For a complete list of supported locale codes, see http://java.sun.com/j2se/1.5.0/docs/guide/intl/locale.doc.html.

### Cb

Sets whether the Forms service generates a form guide with an HTML template. An HTML template is a HTML code that launches the form guide. The default value is False. Select one of these values:

**False:**

The form guide is provided with an HTML template.

**True:**

Only the form guide is provided.

### Guide Style

Sets the name of the cascading style sheet (CSS), such as CobaltBar.css, that the form guide layout uses to provide a customized appearance. For example, can use the following CSS files that you can also export from Designer. (See Designer Help and the Customizing Form Guides using CSS: Missingin action blog entry.)

### Guide Submit All

For example, in your form guide, you can choose to only expose a subset of fields to a user. For example, you can choose to hide some of the values in a hidden panel. Regardless of whether the user modifies the fields, you want all the data submitted, including the data in the hidden panels. The default value is False. Select one of these values:

**False:**

Does not submit all data. Only the data modified by the user is submitted.

**True:**

Submits all data, including hidden panels.

### Inject Form Bridge

Sets whether special JavaScript code is inserted in the form design. The JavaScript code permits the rendered form guide to be used in Workspace. The default value is False. Select one of these values:

**False:**

Does not enable the PDF form for use in Workspace.

**True:**

Enables the PDF form for use in Workspace.

# 21.44. FormsResult

A complex data type that holds the results of Forms service operations. `FormsResult` variables are used to store the results of a Forms service operation, such rendering result, the button that the user clicked to submit the form, and the locale of the form. See the Data items section for complete information.

You can create `FormsResult` variables to save the value of the Forms Result property for each of these operations:

processFormSubmission

renderFormGuide

renderHTMLForm(deprecated)

(Deprecated)renderHTMLForm

renderPDFForm

renderPDFForm(deprecated)

For information about data that can be accessed using Xpath Expressions, see Dataitems.

## Data items

The data items that `FormsResult` variables contain.

### action

A `short` value that indicates the processing state of the operation when it provided the result. The state indicates whether interaction between the Forms service and the client application, such as a web browser, is complete. These values are valid:

**0:**

Interaction is complete and validated XML data is ready to be processed.

**1:**

The data contains calculation results that must be returned to the web browser so that the user can view the results.

**2:**

Calculations and validations must be written to the web browser.

**3:**

(HTML transformations) The current page has changed with results that must be written to the web browser.

**4:**

(HTML transformations) The current page has changed with results that must be written to the web browser.

### attachments

A `list` of `document` values that represent (PDF) file attachments that were part of the rendered form. For information about retrieving data items from a `list` value, see Accessingdata in data collections.

### charSet

A `string` value that represents the character set encoding that was used, such as `ISO-8859-1`, `UTF-8`, or `SHIFT_JIS`. For a complete list of character sets, see http://java.sun.com/j2se/1.5.0/docs/guide/intl/encoding.doc.html.

### clickedBtn

A `string` value that represents an XML architecture Scripting Object Model (SOM) expression that identifies the last button that was clicked a form. A SOM expression references objects, properties, and methods within a Document Object Model (DOM). The following example code is a SOM expression:

```
Untitled[0].main[0].ButtonHello[0]
```

The items `Untitled` and `main` represent branches in the form's DOM hierarchy. The name of the button on the form is `ButtonHello`. For information about the XML architecture, see Adobe XML Forms Architecture.

### contentType

A `string` value that represents the MIME content type of the data that the Forms service generated. These string values are valid:

- text/xml
- text/html
- application/xml
- application/pdf
- application/vnd.adobe.xdp+xml

If the MIME content type is `text/html`, the character set encoding that was used is appended to the value, such as `text/html; charset="ISO-8859-1"`.

### formQuery

A `string` value that represents the filename of the form that was rendered.

### locale

A `string` value that language used to send validation messages to the client application, such as web browsers. For a complete list of supported locale codes, see http://java.sun.com/j2se/1.5.0/docs/guide/intl/locale.doc.html.

### options

A `string` value that represents an ampersand-delimited list of options that were used for the operation.

### outputContent

A `document` value that holds the data that the Forms service generated. Depending on the value of the action data item, the content is sent back to the client application for further processing.

### outputString

A `string` value that represents the data that the Forms service generated. If the data is binary, this value is an empty string.

### outputType

An `int` value that indicates the output type of a form that is rendered as HTML. These values are valid:

**0:**

The form is rendered within `HTML` elements (a complete HTML page).

**1:**

The form is rendered within `body` elements (not a complete HTML page).

### outputXML

A `document` value that holds XML or HTML data modified by client application. It is possible to enhance HTML forms by adding elements not supported by the Forms service. For example, the client application can change an input element to be a password type, which is not a supported element.

### pageCount

A `long` value that holds the total number of pages within the generated output.

### pageNumber

A `long` value that holds the current page number in an HTML form. The page number of the first page is zero. This data item is not relevant form guides or PDF forms.

### transformationID

A `string` value that represents the actual transformation performed. These string values represent what was render operation occurred:

**PDFForm:**

The Forms service rendered a PDF form, which also includes a PDF form with merged data.

**FormGuide:**

The Forms service rendered a form guide.

**AHTML/HTML4/MSDHTML/NoScriptXHTML/StaticHTML/AccessibleXHTML/NoScriptAccessibleX-HTML:**

The Forms rendered an HTML form. The format of the HTML depends on the value of the Transform To property. See the `TransformTo` data type for more information about each format.

**validationErrorsList**

A `document` value that contains validation errors as UTF8-encoded XML. This information is useful to client applications that want to access errors directly after the operation executes, but before any content is returned. The following XML shows the valid format:

```
<validationerrors>
<!-- page level errors -->
<pages />
<!-- field level errors -->
<fields>
<field>
<name>rootsubform[1].FFField1[1]</name>
<message>Mandatory Field</message>
</field>
</fields>
</validationerrors>
```

**XMLData**

A `document` value that holds UTF8-encoded XML data from the Forms service when the Populate XML Data property is set to True. The Populate XML Data property is enabled using the Render Options properties of processFormSubmission operation. The XML data is based on the current processing state of the form and the result of any calculations. The result of a calculation is indicated when the value of the FSAction data item is Calculate. The result of a calculation is not always the same result as the XML data returned after a form is submitted. A form is submitted when the FSAction data item is a value of Submit.

## 21.45. FTP FileInfo

A complex data type that contains information about a file that is retrieved by the Getlist of files operation of the FTP service. Each data item that the variable holds contains information about the file.

For information about data that can be accessed using Xpath Expressions, see Dataitems.

## Data items

The data items that `FTP FileInfo` variables contain.

### directory

A `boolean` value that specifies whether the value specified in the Remote Directory Path property of the Get List Of Files operation is a directory path. A value of `true` indicates it is a directory and `false` indicates it is not a directory.

### directoryName

A `string` value that represents the name of the directory from which the file was retrieved.

### fileName

A `string` value that represents the name of the file that was retrieved.

### fullPath

A `string` value that represents the path of the file that was retrieved.

### hidden

A `boolean` value that indicates whether the file was hidden. A value of `true` indicates the file was hidden and `false` indicates it was not hidden.

### readOnly

A `boolean` value that indicates whether the file was read only. A value of `true` indicates the file was read only and a value of `false` indicates it was not read only.

### size

A `long` value that represents the size of the file that was retrieved, in bytes.

# 21.46. GetUsageRightsResult

A complex data type used by the GetDocument Usage Rights and GetCredential Usage Rights operations of the Acrobat Reader DC extensions service. It represents information about a credential and the individual usage rights associated with it.

For information about data that can be accessed using Xpath Expressions, see Dataitems.

## Data items

The data items that `GetUsageRightsResult` variables contain.

**evaluation**

A `boolean` value that indicates whether the deployment type of the credential is Evaluation and Test.

**intendedUse**

A `string` value that represents the legal intended-use notice of the credential.

**message**

A `string` value that represents the message that appears when the PDF document is opened in Adobe Reader. The message informs users that the document contains Acrobat Reader DC extensions usage rights.

**notAfter**

A `date` value that specifies the date after which the credential is no longer valid.

**notBefore**

A `date` value that specifies the date before which the credential is not valid.

**profile**

A `string` value that represents the profile code of the credential. The profile code is a short description of complete certificate properties (for example, P8).

**rights**

A `rights` value that specifies individual usage rights associated with a PDF document or credential.

**useCount**

An `int` value that specifies the number of times the credential has been used.

## 21.47. Group

A complex data type that represents a user group that is configured in User Manager. This variable type is used for the following purposes:

- To store the results of the FindGroup and FindGroups operations that the User Lookup service provides
- To configure findGroupMembers operations
- To configure the Initial User Selection property of AssignTask operation operations that the User service provides

`Group` variables store attribute values for groups that exist in User Management.

For information about data that can be accessed using Xpath Expressions, see Dataitems.

### Data items

#### canonicalName

A `string` value that represents the canonical name for the group.

#### commonName

A `string` value that represents the common name of the group.

#### domainName

A `string` value that represents the name of the User Manager domain that the group belongs to.

#### principalOid

A `string` value that represents the identifier of the principal object that represents the group. The principalOid uniquely identifies the group in the AEM Forms environment.

RELATED LINKS:

User

User Lookup

## 21.48. HashAlgorithm

A `string` value that represents hash algorithms used to digest the PDF document. `HashAlgorithm` variables are used to configure the Digital Hashing Algorithm property in the CertifyPDF and SignSignature Field operations of the Signature service. These string values are valid:

**SHA1:**

The Secure Hash Algorithm that has a 160-bit hash value.

**SHA256:**

The Secure Hash Algorithm that has a 256-bit hash value.

**SHA384:**

The Secure Hash Algorithm that has a 384-bit hash value.

**SHA512:**

The Secure Hash Algorithm that has a 512-bit hash value.

**RIPEMD160:**

The RACE Integrity Primitives Evaluation Message Digest that has a 160-bit message digest algorithm and is not FIPS-compliant.

For information about configuring default properties, see Datatypespecific settings.

### Datatype specific settings

Sets the hashing algorithm to use to digest the PDF document. Select one of these values:

**SHA1:**

The Secure Hash Algorithm that has a 160-bit hash value.

**SHA256:**

The Secure Hash Algorithm that has a 256-bit hash value.

**SHA384:**

The Secure Hash Algorithm that has a 384-bit hash value.

**SHA512:**

The Secure Hash Algorithm that has a 512-bit hash value.

**RIPEMD160:**

The RACE Integrity Primitives Evaluation Message Digest that has a 160-bit message digest algorithm and is not FIPS-compliant.

## 21.49. HTMLRenderSpec

A complex data type that contains information for configuring the HTML Render Options property of the renderHTMLForm(deprecated) operation that the Forms service provides.

For information about data that can be accessed using Xpath Expressions, see Dataitems.

For information about configuring default properties, see Datatypespecific settings.

### Data items

The data items that `HTMLRenderSpec` variables contain.

#### cacheEnabled

A `boolean` value that specifies whether the Forms service caches a form to improve performance. Each form is cached after it is generated for the first time. On a subsequent render, if the cached form is newer than the form design's timestamp, the form is retrieved from the cache. A value of `true` indicates the form is cached, and `false` indicates the form is always generated anew.

#### charset

A `string` value that represents the character set used to encode the output byte stream. For a complete list of these values, go to http://java.sun.com/j2se/1.5.0/docs/guide/intl/encoding.doc.html.

### customCSSURI

A `string` value that represents the location of a cascading style sheet (CSS) to use instead of the internal CSS that is created during the HTML transformation.

**TIP:** You can use the FormsIVS tool to create a custom CSS. (See the Forms section in AEM Forms Services.*)*

### debugEnabled

A `boolean` value that determines whether debug-level logging is performed. Debug-level logging provides more information in the J2EE application server's log file that is useful for debugging. A value of `true` indicates that debug-level logging is performed, and `false` indicates that the default level of logging is performed. The default level for logging is set on the server.

### digSigCCSURI

A `string` value that represents the URI for a custom style sheet to use for digital signature user interfaces in HTML forms. AEM Forms Server must have access to the location of the URI.

### fontMapURI

A `string` value that represents the URI for font mappings.

### formModel

A `FormModel` value that represents where the form rendering is performed.

### generateTabIndex

A `boolean` value that specifies whether to generate a tab index for fields in the HTML form. A value of `true` specifies that tab indexes are created for each field in the rendered HTML form. A value of `false` specifies that tab indexes are not created for each field in the rendered HTML form.

### HTMLToolbar

A `HTMLToolbar` value that represents the type of toolbar that is rendered for HTML forms.

### locale

A `string` value that represents the locale that the Forms service uses to send validation messages to client applications, such as web browsers. When no value is provided, the default Locale setting from AEM Forms Server is used. The Locale setting is configured in administration console. (See AEM Forms administration help .)

For a complete list of Locale IDs you can use, see http://java.sun.com/j2se/1.5.0/docs/guide/intl/locale.doc.html.

### outputType

A `string` value that represents how a form that is rendered as HTML is displayed. When no value is provided, the default Output Type setting from AEM Forms Server is used. The Output Type setting is configured in administration console. (See AEM Forms administration help.) These values are valid:

**BodyTags:**

The rendered page is wrapped with `<BODY>` tags.

**FullHTMLTags:**

The rendered page is wrapped with `<HTML>` tags.

### pageNumber

An `int` value that represents the initial page number to display in a multipage HTML form.

### standAlone

A `boolean` value that specifies whether the form is rendered without state information. This value is useful when rendering of an interactive form occurs on the server and the form contains JavaScript code that must be executed.

A value of `true` indicates that the JavaScript code in the form design runs on the client. No interaction is required with AEM Forms Server. JavaScript code is embedded in the rendered HTML form. A value of `false` indicates the form is rendered without state information and without embedded JavaScript that runs on the client web browser. The Forms service renders the form after server-side calculations are performed and the results are displayed in the rendered form. Because of the required interaction with the Forms service, the form cannot be used when offline. A value of true indicates that the form is rendered with state information and with embedded JavaScript. The JavaScript code runs on the only on the client and no interaction is required with AEM Forms Server. In addition, the form can be used offline.

### styleGenerationLevel

A `StyleGenerationLevel` value that represents the styling to output for the rendered HTML form. Internal styles are used for spacing and positional CSS styles. Inline styles are used for appearance styles, such as color, font, and background color.

### toolbarURI

A `string` value that represents the URI of custom collateral that is required for a custom toolbar on the form. Custom collateral includes XML, JS, and CSS files. If you are using the default toolbar, then you do not need to supply custom collateral.

### XCIURI

A `string` value that represents the URI of the XCI file used for rendering.

### XMLData

A `boolean` value that indicates whether the Forms service produces the form's XML data based on the service's current processing state. A value of `true` indicates the XML data is based on the current processing state.

## Datatype specific settings

Properties for specifying the default form and rendering options for the HTML form.

### HTML Output Type

Sets whether to wrap the rendered page with Full HTML Tags or Body Tags. The default value is <Use Server Default>. Select one of these values:

**<Use Server Default>:**

Use the default Output type setting configured on AEM Forms Server. The Output type setting is configured in administration console. (See AEM Forms administration help.)

**Full HTML tags:**

The rendered page is wrapped with `<HTML>` tags.

**Body Tags:**

The rendered page is wrapped with `<BODY>` tags.

### Character Set

Sets the character set used to encode the output byte stream. The default value is <Use Server Default>. Select the character set to use or one of these values.

**<Use Server Default>:**

Use the Character Set setting configured on AEM Forms Server. The Character Set setting is configured in administration console. (See AEM Forms administration help.)

**<Use Custom Value>:**

Use a character set not available in the list. After selecting this value, in the text box beside the list, type the canonical name (Java.nio API) of the encoding set to use. For a complete list of character sets, see http://java.sun.com/j2se/1.5.0/docs/guide/intl/encoding.doc.html.

### Start Page Number

Sets the first page number to start to render in a multi-page HTML form. Page numbering is zero-based, which means that the first page is zero. The default value is `0`.

### Character Set

Sets the character set used to encode the output byte stream. The default value is <Use Server Default>. Select either a character set from the list or one of these values.

**<Use Server Default>:**

Use the Character Set setting configured on AEM Forms Server. The Character Set setting is configured in administration console. (See AEM Forms administration help.)

**<Use Custom Value>:**

Use a character set not available in the list. After select this value, in the text box beside the list, type the canonical name (Java.nio API) of the encoding set to use. For a complete list of character sets, see http://java.sun.com/j2se/1.5.0/docs/guide/intl/encoding.doc.html.

### Locale

Sets the language used to send validation messages to client devices, such as web browsers, when an HTML form is rendered. The default value is <Use Server Default>. Select either a language from the list or one of these values.

**<Use Server Default>:**

Use the Locale setting configured on AEM Forms Server. The Locale setting is configured in administration console. (See AEM Forms administration help.)

**<Use Custom Value>:**

Use a locale setting that is not available in the list. After selecting this value, in the text box beside the list, type the Locale ID of the locale code to use. For a complete list of supported locale codes, see http://java.sun.com/j2se/1.5.0/docs/guide/intl/locale.doc.html.

### Cache Form On Server

Sets whether the rendered form is cached on the server to improve performance. Caching forms on the server improves performance for forms that are rendered on the server. Processing instructions embedded in the form design and the Form Rendering Cache Enabled option determines how caching is performed. For information about configuring the Rendering Cache Enabled option in administration console, see ConfiguringForms.The default value is False. Select one of these values:

**False:**

The form is not cached on the server.

**True:**

The form is cached on the server.

### Populate XML Data

Sets whether the XML data is produced from the form based on the form’s current processing state. The default value is False. Select one of these values:

**True:**

Produces XML data based on the current state of the form.

**False:**

Does not produce XML data based on the state of the form.

**Stand Alone Rendition**

Sets whether the form can be rendered without state information. State information is used for rendered forms that require user interaction with the server for submissions. The default is False. Select one of these values:

**False:**

The form is rendered without state information and without embedded JavaScript that runs on the client web browser. The Forms service renders the form after server-side calculations are performed and the results are displayed in the rendered form. Because of the required interaction with the Forms service, the form cannot be used offline.

**True:**

The form is rendered with state information and with embedded JavaScript. The JavaScript code runs on the only on the client and no interaction is required with AEM Forms Server. In addition, the form can be used offline.

**Form Model**

Sets the location where scripts embedded in the form are executed. The default value is <Use Form Template Default>. Select one of these values:

**<Use Form Template Default>:**

The Forms service checks the form design to determine whether to render on the client or server.

**<Client Side>:**

The form is rendered on the client. Do not use scripts that run on the server. When this value is selected, scripts that run on the server generate a warning on AEM Forms Server.

**<Server Side>:**

The form is rendered on the server.

**<Both Server and Client side>:**

The form is rendered on both the server and the client.

**XCI URI**

Sets the URI location of the XCI file to use for rendering. If the root is not specified, the file is assumed to reside in the location where the AEM Forms EAR files are deployed. For example, in a JBoss turnkey installation, you can see default XCI files in the *[Install*

*folder]*/jboss/server/lc_turnkey/svcdata/XMLFormService folder. *[Install folder]* is the location where AEM Forms is installed. For information about creating XCI files, see Designer Help.

### Digital Signature CSS URI

Sets the URI, such as `C:\\customds.css`, for a custom style sheet to use for digital signature user interfaces in HTML forms. AEM Forms Server must have access to the location of the URI.

**NOTE:** To create a custom CSS file, use Forms-IVS tool to generate a custom CSS based for your form design and output type you provide.

### Custom CSS URI

Sets the URI, such as `C:\customcss.css`, for a custom cascading style sheet to use instead of the internal one emitted as part of the HTML form. Review the changes to the custom CSS file when changes are made to the form design.

- A new type of form object is added to the form design, such as button, text, text field, or check box.
- A form object is removed from the form design.
- Moving a form object, such as label or field, from a child subform to its parent or another subform.

AEM Forms Server must have access to the location of the URI.

**NOTE:** To create a custom CSS file, use Forms-IVS tool to generate a custom CSS based for your form design and output type you provide.

### HTML Toolbar

Sets the type of toolbar that is rendered for HTML forms. The default value is Disabled. Select one of these values:

**Disabled:**

The HTML toolbar is disabled.

**Vertical:**

The HTML toolbar appears in a vertical position.

**Horizontal:**

The HTML toolbar appears in a horizontal position.

### HTML Toolbar URI

Sets the URI location of the HTML toolbar resources to include in the rendered HTML transformation.

### Generate Tab Index

Sets whether to create a tab index for each field in the rendered HTML form. When an HTML form with tab indexes is embedded into another HTML page that does not have tab indexes, tabbing inconsistencies can occur. The default value is True. Select one of these values:

**True:**

Tab indexes are created for each field in the rendered HTML form.

**False:**

Tab indexes are not created for each field in the rendered HTML form. Use this option when you embed the HTML form in another HTML form that does not have tab indexes.

**Style Generation Level**

Sets the styling to output for the rendered HTML form. Internal styles are used for spacing and positional CSS styles. Inline styles are used for appearance styles, such as color, font, and background color. The default is Inline And Internal Styles. Select one of these values:

**Inline And Internal Styles:**

Use both inline and internal CSS styles. Internal styles use spacing and positional CSS styles. Inline styles use appearance styles, such as color, font, and background color.

**Only Inline Styles:**

Use only inline CSS styles from the custom CSS file.

**No Styles:**

Use any CSS an inline or internal styling.

# 21.50. HTMLToolbar

A `string` that represents the appearance of an HTML toolbar that is rendered with an HTML form by the Forms service. `HTMLToolbar` variables are used to configure the HTML Toolbar property for renderHTMLForm operation in the Forms service. In addition, `HTMLToolbar` variables are members of HTMLRenderSpec data types.

For information about data that can be accessed using Xpath Expressions, see Dataitems.

## Data items

`HTMLToolbar` data values store one of these string values:

**Disabled:**

The HTML toolbar is disabled.

**Horizontal:**

The HTML toolbar appears in a horizontal position.

**Vertical:**

The HTML toolbar appears in a vertical position.

# 21.51. IdentityDetails

A complex data type that represents certificate profile information. `IdentityDetail` variables are members of `IdentityInformation` data types.

For information about data that can be accessed using Xpath Expressions, see Dataitems.

## Data items

The data items that `IdentityDetails` variables contain.

### certificate

A `CertificateInformation` value that represents information about the certificate.

### commonName

A `string` value that represents the common name that is associated with the certificate.

### country

A `string` value that represents the name of the country where the certificate issuer resides.

### DNQualifier

A `string` value that represents the Distinguished Name (DN) of the entity that signed and issued the certificate.

### email

A `string` value that represents the email address of the entity that signed and issued the certificate.

### generationQualifier

A `string` value that represents the generation qualifier value that is provided by the entity that signed and issued the certificate.

### givenName

A `string` value that represents the name of the entity that signed and issued the certificate.

### initials

A `string` value that represents the initials of the entity that signed and issued the certificate.

### locality

A `string` value that represents the language or locale of the entity that signed and issued the certificate. For a complete list of supported locale codes, see http://java.sun.com/j2se/1.5.0/docs/guide/intl/locale.doc.html.

**organization**

A `string` value that represents the organization name of the entity that signed and issued the certificate.

**organizationalUnit**

A `string` value that represents the organizational unit name of the entity that signed and issued the certificate.

**pseudonym**

A `string` value that represents the alternative name of the entity that signed and issued the certificate.

**serialNumber**

A `string` value that represents the unique serial number provided by the entity that signed and issued the certificate.

**state**

A `string` value that represents the resident state of the entity that signed and issued the certificate.

**subjectName**

A `string` value that represents the name of the entity whose public key the certificate identifies. The name uses the X.500 standard.

**surName**

A `string` value that represents the last name of the signer associated with the certificate.

**title**

A `string` value that represents the title provided to the entity that signed and issued the certificate.

# 21.52. IdentityInformation

Represents basic validation information of a specific identity certificate, such as the path validation and status.

For information about data that can be accessed using Xpath Expressions, see Dataitems.

## Data items

The data items that `IdentityInformation` variables contain.

**paths**

A `CertificatePath` value that represents path information of the certificates used to sign the PDF document.

**policyQualifierList**

A `list` of `string` values that represents policy qualifier strings present in certificates. The certificates are used to validate the identity.

**signerDetails**

An `IdentityDetails` value that represents information about the identity of the person who signed the PDF document.

**status**

An `IdentityStatus` value that represents that status of the signer.

## 21.53. IdentityStatus

A complex data type that represents whether the signer is trusted. `IdentityStatus` variables are members of `CertificatePath` data types. These string values are valid:

**NOTTRUSTED:**

This signer is not trusted or is invalid because the certificate is invalid or the certificate cannot be chained back to a trusted root.

**TRUSTED:**

The signer is trusted because the certificate is both valid and can be chained back to a trusted root.

**UNKNOWN:**

This verification of the signer cannot be performed.

## 21.54. IGetLinkedLCAssetsLocationResult

A complex data type that stores the results of the Get Linked LC Assets Location operation in the following services:

ContentRepositoryConnector forEMCDocumentum

ContentRepository Connector for IBM FileNet

ContentRepositoryConnector for IBMContentManager

For information about data that can be accessed using Xpath Expressions, see Dataitems.

## Data items

The data items that `IGetLinkedLCAssetsLocationResult` variable contains. The value in the linkedLCAssetsFolderPath item is a combination of the values in the linkedLCAssetsName and linkedLCAssetsURL. For example, if the form name is Form.xdp, and the form is located in folder /Docbase/forms/, then the form template path in the repository is /Docbase/forms/Form.xdp.

### linkedLCAssetsFolderPath

A `string` value that specifies the complete path to the form template in the AEM forms repository.

### linkedLCAssetsName

A `string` value that specifies the name of the form template.

### linkedLCAssetsURL

A `string` value that specifies the path to the folder that contains the form template in the AEM Forms repository.

# 21.55. ILoginSettings.LoginMode

A `string` value that contains the login mode to use to log in to either an EMC Documentum or IBM FileNet repository.

## Literal value

For Documentum, one of these values:

**Use Credentials from process context:**

Uses the AEM Forms User Management credential from the process context for the Documentum authentication.

**Use User Credentials:**

Uses the User Name and Password properties to authenticate with Documentum. Ensure that the user name is valid for the repository specified in the Repository Name property in the Relationship Information property group for this operation.

**Use Documentum Login Ticket:**

Uses the Documentum Login Ticket and User Name properties to authenticate with Documentum. This login ticket must be valid in the current Documentum Foundation Class (DFC) session, and the user name must be set to the correct user of the DFC session.

For FileNet, one of these values:

**Use Credentials from process context:**

Uses the AEM Forms User Management credential from the process context for the FileNet authentication.

**Use User Credentials:**

Uses the User Name and Password properties to authenticate with FileNet. Ensure that the user name is valid for the FileNet Content Engine specified in administration console.

**Use FileNet Credentials Token:**

Uses the FileNet Credentials Token property to authenticate with the FileNet Content Engine. This login token must be valid in the current FileNet session.

# 21.56. InitiatorTO

Internal use only.

# 21.57. int

Holds an integer value (32-bit twos complement).

## Literal value

Any number between `-2147483648` and `2147483647`

## Example

```
153
```

# 21.58. lc7form

Do not use.

Replaces the form data type that was used in AEM Forms Workflow 7.x. This variable type is provided for backwards compatibility.

A complex data type that holds the following information:

- Form data (XDP data)
- Selected action
- Action choices
- Design-time URL to the form template
- Run-time URL to the form template

# 21.59. list

Holds an ordered collection of same-typed data items.

Items in `list` variables are indexed. In XPath expressions, the index is one-based so that the first item in the list has an index of one. (See Accessingdata in data collections.)

# 21.60. Locale

A complex data type used to specify a language and geographic region. `Locale` variables are members of the `Owner` data type.

For information about data that can be accessed using Xpath Expressions, see Dataitems.

## Data items

The data items that locale variables contain.

### country

A `string` value that represents the two-letter country or region code for the locale.

### displayCountry

A `string` value that represents a name for a locale's country that is appropriate for display to the user.

### display Language

A `string` value that represents a name for a locale's language that is appropriate for display to the user.

### displayName

A `string` value that represents a name for the locale that is appropriate for display to the user.

### displayVariant

A `string` value that represents a name for the locale's variant code that is appropriate for display to the user.

### ISO3Country

A `string` value that represents the three-letter country code for the locale.

### ISO3Language

A `string` value that represents the three-letter language code for the locale.

**language**

A `string` value that represents the language code for the locale.

**variant**

A `string` value that represents the variant code for the locale. The value provides additional functionality or customization (for example, to specify a specific operating system or browser).

# 21.61. LoginMode

Specifies the mode to log in to an IBM Content Manager connector.

These values are valid:

**INVOCATION_CONTEXT:**

Use credentials provided by the process that invokes the operation.

**USER_CREDENTIALS:**

Use credentials provided by the user.

For information about configuring default properties, see Datatypespecific settings.

## Datatype specific settings

Default properties for logging in to an IBM Content Manager connector.

**Use Credentials From Process Context:**

The AEM Forms User Management credential from the process context is used for the IBM Content Manager authentication.

**Use User Credentials:**

The User Name and Password properties are used to authenticate with IBM Content Manager.

# 21.62. loginSettings

Specifies the values to login to SharePoint repository.

## Data items

The data items that `loginSettings` contains.

### loginMode

A `string` value that specifies how the operation will be authenticated with SharePoint server. Valid values are INVOCATION_CONTEXT (Use Credentials from process context) and USER_CREDENTIALS (Use User Credentials).

### hostName

A `string` value that specifies the hostname of the machine which hosts Microsoft SharePoint. You can also use the Hostname field to connect with any web application running on a port number other than 80 by specifying the hostname and port number in the format `[hostname]:[port number]`. This field is only used when Login Mode is set to 'Use User Credentials'. No default is provided.

### userName

(Optional) A `string` value that specifies the user name to be used to connect with the SharePoint server. If the user account is of Windows local users or users configured using Forms Authentication on the SharePoint site, only the user name is required. However, domain users for Windows Authentication enabled sites must provide both the domain name and login name in the format `domainName\LoginName`. This field is only used when the Login Mode is set to 'Use User Credentials'. No default is provided.

### password

(Optional) A `string` value that specifies the password to be used to connect with the SharePoint server. This field is only used when Login Mode is set to 'Use User Credentials'. No default is provided.

### domainName

(Optional) A `string` value that specifies the domain in which the SharePoint server is present. This field is only used when Login Mode is set to 'Use User Credentials'.

# 21.63. long

Holds a long integer value (4-bit twos complement).

## Literal value

Any number between `-9223372036854775808` and `+9223372036854775807`

## Example

```
54356789
```

# 21.64. map

Holds a keyed collection of same-typed data items.

Items in `map` variables are key-based. Specific data items can be accessed using XPath expressions that use the key associated with the data item. (See Accessing datain data collections.)

**NOTE:** The order of data items in `map` variables can change.

# 21.65. MessageFormatEnum

A `string` value that represents the format that the SendWith Document operation of the Email service should use to format the message. These values are valid:

- `TEXT` (plain text)
- `HTML`

# 21.66. MDPPermissions

A `string` value that represents the type of changes that an end user can perform on a PDF document without invalidating the certification signature. It is used in the CertifyPDF operation of the Signature service. These string values are valid:

**FormChanges:**

The end user is permitted to fill in the form, instantiate page templates, and sign the form.

**AnnotationFormChanges:**

The end user is permitted to fill in the form, instantiate page templates, sign the form, and create annotations, deletions, and modifications.

**NoChanges:**

The end user is not permitted to change the form. Any change invalidates the signature.

For information about configuring default properties, see Datatypespecific settings.

## Datatype specific settings

Sets the types of changes allowed on a PDF document without making the certification signature invalid. Select one of these values:

**No Changes Allowed:**

No changes to the document are permitted. Any change invalidates the signature.

**Form Fill-in and Digital Signatures:**

Permitted changes include filling in forms, instantiating page templates, and signing.

**Annotations, Form Fill-in, and Digital Signatures:**

Permitted changes include filling in forms and instantiating page templates, as well as creating, deleting, and modifying annotations.

# 21.67. ModeratorTO

Internal use only.

# 21.68. object

Do not use.

This data type was used in AEM Forms Workflow 7.x. It is provided for backwards compatibility.

Holds a serialized Java object. You can use `object` variables to store any type of data. `object` variables are useful in the following situations:

- When you need to save the data that a service operation returns and you do not know the data type.
- When you need to store data of different types in a `list` or `map` variable. Configure the `list` or `map` variable's Subtype property to be object. (See list or map.)

# 21.69. OCSPOptionSpec

A complex data type that represents settings for the Online Certificate Status Protocol (OCSP) revocation checking.

`OCSPOptionSpec` variables are used in the following operations of the Signature service:

CertifyPDF

SignSignature Field

VerifyPDF Signature

VerifyPDF Signature (deprecated)

VerifyXML Signature

For information about data that can be accessed using Xpath Expressions, see Dataitems.

For information about configuring default properties, see Datatypespecific settings.

## Data items

The data items that `OCSPOptionSpec` variables contain.

### allowOCSPNoCheck

A `boolean` value that indicates whether the OCSPNoCheck extension is permitted in the response signing certificate. An OCSPNoCheck extension can be added to the OCSP certificate to prevent validation loops. A value of `true` indicates that the OCSPNoCheck extension is allowed. The default value of `false` indicates that the OCSPNoCheck extension cannot be present in the certificate.

### doSignRequest

A `boolean` value that indicates whether to sign the request. A value of `true` indicates that the request must be signed. The default value of `false` indicates that a signed request is not required.

### goOnline

A `boolean` value that indicates whether to access the network to retrieve OCSP information. A value of `true` indicates that the OCSP checking is performed by accessing the network. The default value of `false` indicates that checking is performed by accessing embedded and cached OCSP responses. This type of checking reduces the amount of network traffic.

### ignoreValidityDates

A `boolean` value that indicates whether to ignore the thisUpdate and nextUpdate time values of the OCSP server response. The thisUpdate and nextUpdate times are external sources that are retrieved through HTTP or LDAP and can be different for each revocation information. A value of `true` indicates that the validity dates are ignored. The default value of `false` indicates that validity dates are used, which can negatively affect the response validity.

### maxClockSkew

A `long` value that indicates, in minutes, the maximum allowed skew in OCSP server response time and local time. The default value is `5`.

### ocspServerURL

A `string` value that specifies the URL for the OCSP server.

### requestSignerCredentialAlias

A `string` value that represents the alias that corresponds to the credential that is used to sign the PDF document.

### RequireOCSPCertHash

A `string` value that indicates whether a certificate public key hash extension is required in the OCSP (Online Certificate Status Protocol) responses. This extension is required in the case of SigQ validation. SigQ compliance requires the CertHash extension to be in the OCSP responder certificate. Set this property to `true` only when processing for SigQ compliance and supported OCSP responders. A value of `true` indicates that the CertHash extension must be in the OCSP responder. The default value of `false` indicates that the CertHash extension does not need to be in the OCSP responder certificate.

### responseFreshness

An `int` value that specifies the maximum time validity (in minutes) of the preconstructed OCSP response. The default value is `52600` (one year).

### revocationCheckStyle

A `string` value that specifies the type of revocation-checking that is performed when verifying a signature in a PDF document.

These string values are valid:

**AlwaysCheck:**

Checks for revocation of all certificates.

**BestEffort:**

Checks for revocation of all certificates when possible.

**CheckIfAvailable:**

Checks for revocation of all certificates only when revocation information is available.

**NoCheck:**

Does not check for revocation.

The default value is `CheckIfAvailable`.

### sendNonce

A `boolean` value that indicates whether a nonce is sent with the request. A *nonce* is a parameter that can be a timestamp, a visit counter on a web page, or a special marker. The parameter is intended to limit or prevent an unauthorized replay or reproduction of a file. The default value of `true` indicates that a nonce is sent with the request and a value of `false` indicates that a nonce is not in the request.

### URLtoConsultOption

A `string`value (with a finite list of valid values) that represents the type of OCSP servers and the order to use them when performing the revocation check. The default value is `UseAIAInCert`.

These values are valid:

**LocalURL:**

Use the locally configured URL.

**UseAIAInCert:**

Use the URL of an online certificate status protocol server specified in the Authority Information Access (AIA) extension in the certificate. The AIA extension is used to identify how to access certificate authority (CA) information and services for the issuer of the certificate.

**UseAIAIfPresentElseLocal:**

Use the URL of the OCSP server specified in the AIA extension in the certificate if present. If the AIA extension is not present in the certificate, use the URL configured in the OCSP Server URL.

**UseAIAInSignerCert:**

Use the URL of the OCSP server specified in the AIA extension in the signer certificate.

## Datatype specific settings

Properties for the Online Certificate Status Protocol (OCSP).

### URL to Consult Option

Sets the list and order of the OCSP servers used to perform the revocation check. The default value is UseAIAInCert. Select one of these values:

**UseAIAInCert:**

Use the URL of an online certificate status protocol server specified in the Authority Information Access (AIA) extension in the certificate. The AIA extension is used to identify how to access certificate authority (CA) information and services for the issuer of the certificate.

**LocalURL:**

Use the specified URL for the OCSP server specified in the OCSP Server URL option.

**UseAIAIfPresentElseLocal:**

Use the URL of the OCSP server specified in the AIA extension in the certificate if present. If the AIA extension is not present in the certificate, use the URL configured in the OCSP Server URL.

**UseAIAInSignerCert:**

Use the URL of the OCSP server specified in the AIA extension in the signer certificate.

### OCSP Server URL

Sets the URL of the configured OCSP server. The value is only used when the LocalURL or UseAIAIfPresentElseLocal values are in URL To Consult Option.

### Revocation Check Style

Sets the revocation-checking style used for verifying the trust status of the CRL provider's certificate from its observed revocation status. The default value is CheckIfAvailable. Select one of these values:

**NoCheck:**

Does not check for revocation.

**BestEffort:**

Checks for revocation of all certificates when possible.

**CheckIfAvailable:**

Checks for revocation of all certificates only when revocation information is available.

**AlwaysCheck:**

Checks for revocation of all certificates.

### Max Clock Skew Time (minutes)

Sets the maximum allowed skew, in minutes, between response time and local time. Valid skew times are `0` - `2147483647` min. The default value is `5` min.

### Response Freshness Time (minutes)

Sets the maximum time, in minutes, for which a preconstructed OCSP response is considered valid. Valid response freshness times are `1`- `2147483647` min. The default value is `525600` min. (one year).

### Send Nonce

Select this option to send a nonce is with the OCSP request. A *nonce* is a parameter that varies with time. These parameters can be a timestamp, a visit counter on a web page, or a special marker. The parameter is intended to limit or prevent the unauthorized replay or reproduction of a file. When the option deselected, a nonce is not sent with the request. By default, the option is selected.

### Sign OCSP Request

Select this option to specify that the OCSP request must be signed. When the option is deselected, the OCSP request does not need be signed. By default, the option is deselected.

### Request Signer Credential Alias

Sets the credential alias used for signing the OCSP request when signing is enabled.

### Go Online for OCSP

Select this option to access embedded and cached OCSP responses on AEM Forms Server. The network can be accessed to retrieve OCSP information for OCSP checking. Accessing OCSP responses on the server helps to reduce the amount of network traffic generated due to OCSP checking. When the option deselected, OCSP checking is performed by accessing AEM Forms Server. By default, the option is selected.

### Ignore Validity Dates

Select this option to use the OCSP response thisUpdate and nextUpdate times. Ignoring the response's thisUpdate and nextUpdate times prevents any negative effect on response validity. The thisUpdate and nextUpdate times are retrieved from external sources by using HTTP or LDAP and can be different for each revocation information. When the option is deselected, the thisUpdate and nextUpdate times are ignored. By default, the option deselected.

### Allow OCSP NoCheck Extension

Select this option to allow an OCSPNoCheck extension in the response signing certificate. An OCSPNo-Check extension can be present in the OCSP Responder's certificate to prevent infinite loops from occurring during the validation process. When the option deselected, the OCSPNoCheck extension is not allowed. By default, the option is selected.

### Require OCSP ISIS-MTT CertHash Extension

Select this option to specify that certificate public key hash (CertHash) extensions must be present in OCSP responses. This extension is required for SigQ validation. SigQ compliance requires the CertHash extension to be in the OCSP responder certificate. Select this option when processing for SigQ compliance and supported OCSP responders. When the option is deselected, the CertHash extension presence in the OCSP response is not required. By default, the option is deselected.

# 21.70. OutputJog

A `string` value that represents Jog XCI values. These values are used when output pages are jogged (physically shifted in the output tray). `OutputJob` variables are used to configure the Output Jog property in the generatePrintedOutput operation of the Output service.

For information about data that can be accessed using Xpath Expressions, see Dataitems.

For information about configuring default properties, see Datatypespecific settings.

## Data items

`OutputJog` data values store one of these string values:

**usePrinterSetting:**

Use the jog option setting configured on the printer.

**none:**

Do not use jogging.

**pageSet:**

Use jog each time a set of pages is printed.

## Datatype specific settings

Property that specifies whether to use jogging.

### Default Value

Sets whether to use jogging on the printer. Select one of these values:

**usePrinterSetting**

Use the jog option setting configured on the printer.

**none:**

Do not use jogging.

**pageSet:**

Use jog each time a set of pages is printed.

# 21.71. OutputResult

A complex data type that stores the results of an Output service operation. `OutputResult` variables are used to store the results in the generatePDFOutput(deprecated), generatePDFOutput, generatePrintedOutput(deprecated), and generatePrintedOutput operations of the Output service.

For information about data that can be accessed using Xpath Expressions, see Dataitems.

## Data items

The data items that `OutputResult` variables contain.

**generateDoc**

A `document` value that represents the PDF document that the Output service generates.

**metaDataDoc**

A `document` value that contains the metadata for the generated PDF document.

**recordLevelMetaDataList**

A `list` of `string` values that represent the metadata used to create the metadata file.

**statusDoc**

A `document` value that represents the status of the operation. If an error occurs and the generatePDFOutput(deprecated) or generatePrintedOutput(deprecated) operations were used, the error message is stored in this value. When the generatePDFOutput and generatePrintedOutput operations are used, the value returned is `null` and the error message is available in the application server log file.

# 21.72. Output Type

A simple data type that stores a string value that represents a type of HTML output. This data type is used to configure HTML Output Type properties of renderHTMLForm operations (Forms service).

The following string values are valid:

- **FullHTMLTags:** A complete HTML page, wrapped with `<HTML>` tags.
- **BodyTags:**An HTML page wrapped with `<BODY>` tags.

You can set the default value of Output Type variables using the Datatype Specific Settings.

# 21.73. OutputType

The output type value that specifies how a form that is rendered as HTML is displayed by the Forms service. The following `string` values are valid:

**Body Tags:**

The HTML form is rendered within body tags (not a complete HTML page).

**Full HTML Tags:**

The HTML form is rendered within full HTML tags (a complete HTML page).

# 21.74. Owner

A complex data type used to specify a user who is an owner (for example, the owner of a policy).

For information about data that can be accessed using Xpath Expressions, see Dataitems.

## Data items

The data items that owner variables contain.

**businessCalendarKey**

A `string` value that represents the business Calendar Key.

**disabled**

A `boolean` value that specifies whether the user is disabled.

**familyName**

A `string` value that represents the family name.

**givenName**

A `string` value that represents the given name.

**initials**

A `string` value that represents the initials.

**locale**

A `Locale` value that represents the locale of the owner.

**postalAddress**

A `string` value that represents the postal address.

**telephoneNumber**

A `string` value that represents the telephone number.

**timezone**

A `Timezone` value that represents the time zone of the user.

**userid**

A `string` value that represents the user identifier.

# 21.75. Pagination

A `string` value that represents printer pagination settings. Pagination is one of the XCI values used by the Output service. `Pagination` variables are used to configure Duplex Printing property in the generatePrintedOutput operation of the Output service.

For information about data that can be accessed using Xpath Expressions, see Dataitems.

For information about configuring default properties, see Datatypespecific settings.

## Data items

`Pagination` data values store one of these string values:

**duplexLongEdge:**

Use two-sided printing and print using long edge pagination.

**duplexShortEdge:**

Use two-sided printing and print using short edge pagination.

**simplex:**

Use single-sided printing.

## Datatype specific settings

Property for specifying the default value to use for pagination.

**Default Value**

The pagination to use for printing. Select one of these values:

**duplexLongEdge:**

Use two-sided printing and print using long edge pagination.

**duplexShortEdge:**

Use two-sided printing and print using short edge pagination.

**simplex:**

Use single-sided printing.

# 21.76. ParticipantTO

A complex data type that represents a participant in a review, such as the initiator, reviewer, moderator, or approver. Used with the Review, Commenting, and Approval building block, which is installed with the Managed Review & Approval Solution Accelerator.

For information about data that can be accessed using XPath Expressions, see Dataitems.

## Data items

The data items that ParticipantTO values contain.

**additionalMetadata**

A `string` value that represents additional information that can be provided for a participant.

**completedBy**

A `string` value that represents the user who completed the review.

**completedFromIP**

A `string` value that represents the IP address of the client machine from which the review was completed.

**disposition**

A `string` value that represents a custom status for a participant, such as a reviewer or approver.

**finalComments**

A `string` value that represents the final comments of the participant after reviewing or approving the document.

**id**

Internal use only.

**reviewContext**

A ReviewContextTO value that represents the instance of the review.

**status**

A `string` value that represents the status of the participant.

**umOid**

A `string` value that represents the User Management identifier for the participant.

# 21.77. Password Encryption Option Spec

A complex data type that is used as an input value for the PasswordEncrypt PDF operation that the Encryption service provides. This variable specifies the PDF encryption options for compatibility with different versions of Adobe Acrobat, the PDF document resources to encrypt, the password to modify the permissions, and the password to open the PDF document.

For information about data that can be accessed using Xpath Expressions, see Dataitems.

For information about configuring default properties, see Datatypespecific settings.

## Data items

The data items that `Password Encryption Options Spec` variables contain.

**compatibility**

A `Password Encryption Compatibility` value that specifies the earliest version of Acrobat that is compatible with the encryption used. These string values are valid:

**ACRO_3:**

Compatible with Acrobat 3 and later releases that provides 40-bit encryption.

**ACRO_5:**

Compatible with Acrobat 5 and later releases that provide 128-bit encryption.

**ACRO_6:**

Compatible with Acrobat 6 and later releases that provide 128-bit encryption and also provide a finer granularity of control over what operations are permitted.

**ACRO_7:**

Compatible with Acrobat 7 and later releases that provide 128-bit Advanced Encryption Standard (AES).

**ACRO_9:**

Compatible with Acrobat 9 and later releases that provide 256-bit AES.

**documentOpenPassword**

A `string` value that represents the password to open the encrypted PDF document.

**encryptOption**

A `Password Encryption Option` value that specifies the parts of the PDF document to encrypt. These string values are valid:

**ALL:**

The entire PDF document is encrypted.

**ALL_EXCEPT_METADATA:**

The entire PDF document except for the metadata is encrypted.

**ATTACHMENTS_ONLY:**

Only the PDF document attachments are encrypted.

**permissionPassword**

A `string` value that represents the password to modify the permissions or remove encryption on a PDF document.

**permissionRequested**

A `Password Encryption Permission` value that represents the permissions used to secure a document. These string values are valid:

**PASSWORD_ALL_PERM:**

All the permissions mentioned in this list.

**PASSWORD_EDIT_ADD:**

The permission that enables users to add form data.

**PASSWORD_EDIT_ASSEMBLE:**

The permission that enables users to assemble PDF documents.

**PASSWORD_EDIT_COPY:**

The permission that enables users to copy form data.

**PASSWORD_EDIT_EXTRACT:**

The permission that enables users to extract data from forms.

**PASSWORD_EDIT_FORM_FILL:**

The permission that enables users to fill in forms.

**PASSWORD_EDIT_MODIFY:**

The permission that enables the users to edit the PDF document.

**PASSWORD_PRINT_HIGH:**

High-resolution printing of documents is allowed.

**PASSWORD_PRINT_LOW:**

Low-resolution printing of documents is allowed.

**PASSWORD_PRINT_NONE:**

Does not allow printing of documents.

## Datatype specific settings

Properties for setting the default values of the variable.

### Compatibility

The earliest version of Acrobat that is compatible with the encryption method that is used:

**Acrobat 3.0 and later:**

Compatible with Acrobat 3 and later releases that provide 40-bit encryption.

**Acrobat 5.0 and later:**

Compatible with Acrobat 5 and later releases that provide 128-bit encryption.

**Acrobat 6.0 and later:**

Compatible with Acrobat 6 and later releases that provide 128-bit encryption and also provide a finer granularity of control over what operations are permitted.

**Acrobat 7.0 and later:**

Compatible with Acrobat 7 and later releases that provide 128-bit Advanced Encryption Standard (AES).

**Acrobat 9.0 and later:**

Compatible with Acrobat 9 and later releases that provide 256-bit AES.

**All Document Contents**

Select this option to encrypt the entire PDF document.

**All Contents Except Metadata**

Select this option to encrypt the entire PDF document except for the metadata.

**Attachments Only**

Select this option to encrypt only the PDF document attachments.

**Document Open Password**

The password required to open the encrypted PDF document.

**Restrict Edit Of Security Settings**

Select this option to prevent changes being made to security settings.

**Permissions Password**

The password required to change security settings.

**Printing Allowed**

The type of printing that is allowed on the PDF document:

**None:**
No printing is allowed.

**High Resolution:**
High-resolution printing of documents is allowed.

**Low Resolution (150 dpi):**
Low-resolution printing of documents is allowed.

**Changes Allowed**

Select the types of changes that can be performed on the PDF document.

## 21.78. Password Encryption Type

A `string` value that represents the types of security that are used to secure a PDF document. These values are valid:

**CERTIFICATE:**
The PDF document is encrypted with a certificate.

**NONE:**

The PDF document is not encrypted.

**OTHER:**

The PDF document is encrypted another form of encryption.

**PASSWORD:**

The PDF document is encrypted with a password.

**POLICY_SERVER:**

The PDF document is protected with a policy created by the Rights Management service.

# 21.79. PathValidationOptionSpec

A complex data type that represents RFC3280-related path validation options. It is used by the VerifyPDF Signature and VerifyPDF Signature (deprecated) operations in the Signature service.

For information about data that can be accessed using Xpath Expressions, see Dataitems.

For information about configuring default properties, see Datatypespecific settings.

## Data items

The data items that `PathValidationOptionSpec` variables contain.

### anyPolicyInhibit

A `boolean` value that indicates whether any policy can be processed if it is included in the certificate. A value of `true` indicates that any policy is not processed. The default value of `false` indicates that any policy can be processed.

### checkAllPaths

A `boolean` value that specifies whether all paths to a trust anchor are checked for validity. A value of `true` indicates that all paths are checked. The default value of `false` indicates not to validate the paths.

### checkCABasicConstraints

A `boolean` value that indicates whether the CA Basic Constraints certificate extension must be present for CA certificates. For example, earlier versions of some certificates are not compliant with RFC 3280 and do not contain the basic constraints extension. The default value of `true` indicates that CA Basic Constraints certificate extension is required, and `false` indicates that the certificate extension is not required.

### explicitPolicy

A `boolean` value that indicates whether the path must be valid for at least one of the certificate policies in the user's initial policy set. A value of `true` indicates that there must be at least one valid certificate policy path. The default value of `false` indicates that no valid path is required.

### followURIsInAIA

A `boolean` value that indicates whether to follow any URIs specified in the certificate's Authority Information Access (AIA) extension for path discovery. The AIA extension specifies where to find up-to-date certificates. A value of `true` indicates to follow URIs in the certificate’s AIA extension. The default value of `false` indicates not to follow URIs.

### LDAPServer

A `string` value that specifies the Lightweight Directory Access Protocol (LDAP) server that is used to retrieve certificate revocation list (CRL) information. The LDAP server searches for CRL information by using Distinguished Name (DN) according to the rules specified in RFC3280, section 4.2.1.14.

### policyMappingInhibit

A `boolean` value that indicates whether policy mapping is allowed in the certification path. A value of `true` means that policy mapping is not allowed. The default value of `false` means that policy mapping is allowed.

### requireValidSigForChaining

A `boolean` value that indicates whether chains can be built with invalid signatures. A value of `true` indicates that the chain is not built if an invalid signature is encountered. The default value of `false`indicates that invalid signatures are ignored when building the chain.

## Datatype specific settings

Properties for specifying the path validation options.

### Require Explicit Policy

Select this option to specify that the path must be valid for at least one of the certificate policies in the user initial policy set. When this option is deselected, the path validity is not required. By default, the option is deselected.

### Inhibit ANY Policy

Select this option to specify that a policy object identifier (OID) must be processed if it is included in a certificate. When deselected, any policy can be selected. By default, the option is deselected.

### Check All Paths

Select this option to require all paths to a trust anchor must be validated. When this option is deselected, all paths to a trust anchor are not validated. By default, the option is deselected.

### Inhibit Policy Mapping

Determines whether policy mapping is allowed in the certification path. If selected, policy mapping is allowed. This option is not selected by default.

### LDAP Server

Sets the URL or path of the Lightweight Directory Access Protocol (LDAP) server used to retrieve information about the certificate revocation list (CRL). The LDAP server searches for CRL information using the distinguished name (DN) according to the rules specified in RFC3280, section 4.2.1.14. For example, you can type `www.ldap.com` for the URL or `ldap://ssl.ldap.com:200` for the path and port.

### Follow URIs in Certificate AIA

Select this option to specify to follow any URIs specified in the certificate's Authority Information Access (AIA) extension for path discovery. The AIA extension specifies where to find up-to-date certificates. When this option is deselected, no URIs are processed in the AIA extension from the certificate. By default, the option is deselected.

### Basic Constraints Extension required in CA Certificates

Select this option to specify that the certificate authority (CA) Basic Constraints certificate extension must be present for CA certificates. Some early German certified root certificates (7 and earlier) are not compliant to RFC 3280 and do not contain the basic constraint extension. If it is known that a user's EE certificate chains up to such a German root, deselect this option. When this option is deselected, the presence of the CA Basic Constraints certificate in CA certificates is not required. By default, the value is selected.

### Require Valid Certificate Signature During chain building

Select this option to require that all Digital Signature Algorithm (DSA) signatures on certificates be valid before a chain is built. For example, in a chain CA > ICA > EE where the signature for EE is not valid, the chain building stops at ICA. EEs are not included in the chain. When this option is deselected, the entire chain is built regardless of whether an invalid DSA signature is encountered. By default, the option is deselected.

## 21.80. PDFAConformance

A `string` value that represents the PDF/A conformance level as specified by the *Document management - Electronic document file format for long-term preservation* specification (ISO-19005-1:2005). The *conformance level* indicates how a PDF document adheres to the electronic document preservation requirements. `PDFAConformance` variables are used to configure PDF/A Conformance properties in the transformPDF and generatePDFOutput operations of the Output service.

For information about data that can be accessed using Xpath Expressions, see Dataitems.

For information about configuring default properties, see Datatypespecific settings.

### Data items

`PDFAConformance` data values store one of these string values:

**A:**

**B:**

Level B conformance specifies minimal compliance with ISO 19005. The PDF file is generated with all fonts embedded, the appropriate PDF bounding boxes specified, and colors as CMYK, spot colors, or both. Compliant files must contain information describing the printing condition they are prepared for. PDF files created with PDF/X-1a compliance can be opened in Acrobat 4.0 and Adobe Reader 4.0 and later.

### Datatype specific settings

Property for specifying the PDF/A conformance level to use.

#### Default Value

The compliance level for a PDF/A document output. Select one of these values:

**A:**

Level A conformance specifies complete conformance with ISO 19005. The PDF file is generated using PDF 1.4 and all colors are converted to either CMYK or RGB. These PDF files can be opened in Acrobat and Adobe Reader 5.0 and later.

**B:**

Level B conformance specifies minimal compliance with ISO 19005. The PDF file is generated with all fonts embedded, the appropriate PDF bounding boxes specified, and colors as CMYK, spot colors, or both. Compliant files must contain information describing the printing condition they are prepared for. PDF files created with PDF/X-1a compliance can be opened in Acrobat 4.0 and Adobe Reader 4.0 and later.

## 21.81. PDFAConversionOptionSpec

A complex data type that represents the conversion options for converting a PDF document to PDF/A. This data type is used as an input property for the Convert toPDF/A operation that the DocConverter service provides.

For information about data that can be accessed using Xpath Expressions, see Dataitems.

For information about configuring default properties, see Datatypespecific settings.

## Data items

The data items that `PDFAConversionOptionSpec` variables contain.

### colorSpace

A `string` value that represents the color space management scheme to use. The default value is `S_RGB`. These values are valid:

**S_RGB:**

Standard RGB IEC61966-2.1

**COATED_FOGRA27:**

Europe ISO Coated FOGRA27

**JAPAN_COLOR_COATED:**

Japan Color 2001 Coated

**SWOP:**

U.S. Web Coated (SWOP) v2

### compliance

A `string` value that represents the compliance level for converting to PDF/A. The default value is `PDFA_1B`, which is the minimal level for documents scanned from paper or microfiche sources.

### jobLevel

A `string` value that represents the log level to set. The default is INFO. These values are valid:

**OFF:**

Turn off logging of messages on the server.

**SEVERE:**

Log only messages on the server that indicate a serious failure.

**WARNING:**

Log only messages that indicate a potential problem.

**INFO:**

Log only informational messages.

**CONFIG:**

Log only static configuration messages.

**FINE:**

Log tracing information.

**FINER:**

Log fairly detailed tracing information.

**FINEST:**

Log highly detailed tracing information.

**NOTE:** The use of `FINE`, `FINER`, and `FINEST` will negatively affect performance on the server. Use for debugging purposes only.

**resultLevel**

A `string` value that represents the level of results provided when the operation runs. The default value is `PASS_FAIL`. These values are valid:

**PASS_FAIL:**

A report that contains the following information about the operation result:

- Whether the operation was successful
- Whether the PDF document is PDF/A-compliant
- Whether certified digital signatures are permitted

**SUMMARY:**

A report contains the following information about the operation result:

- Whether the operation was successful
- Whether the PDF document is PDF/A-compliant
- Whether certified digital signatures are permitted
- A summarized count of any validation errors that occurred

**DETAILED:**

A report contains the following information about the operation result:

- Whether the operation was successful
- Whether the PDF document is PDF/A-compliant
- Whether certified digital signatures are permitted
- A list of each error that occurred with specific information about each error

**signatures**

A `string` value that represents whether to archive a digital signature. The default value is `ARCHIVE_AS_NEEDED`. These values are valid:

**ARCHIVE_AS_NEEDED:**

Signatures are archived only when the signature is no longer maintainable.

**ARCHIVE_ALWAYS:**

Signatures are archived regardless of their validity to remain in the resulting PDF/A.

### Datatype specific settings

Properties for PDF conversion.

#### Compliance

The compliance level for converting to PDF/A. The default value is `PDFA_1B`.

#### Result Level

The level of results provided when the operation runs. The default value is `PASS_FAIL`.

#### Signatures

The type of archive required for a digital signature. The default value is `ARCHIVE_AS_NEEDED`.

#### Color Space

The color space management scheme to use. The default value is `S_RGB`.

#### Verify Conversion

Determines whether to verify the conversion result. If selected, the result is verified. This option is selected by default.

#### Job Log Level

The logging level to set. The default is `INFO`.

## 21.82. PDFAConversionResult

A complex data type that stores the results of the Convertto PDF/A operation that the DocConverter service provides.

For information about data that can be accessed using Xpath Expressions, see Dataitems.

### Data items

The data items that `PDFAConversionResult` variables contain.

**conversionLog**

A `document` value that represents the conversion results.

**isPDFA**

A `boolean` value that indicates whether the conversion completed successfully. A value of `true` means that a PDF was successfully converted to PDF/A, and `false` means that the conversion failed.

**jobLog**

A `document` value that represents the results of an operation.

**PDFADocument**

A `document` value that stores the PDF/A document that was generated.

# 21.83. PDFARevisionNumber

A `string` value that represents the PDF/A revision number as specified by *Document management - Electronic document file format for long-term preservation* specification (ISO-19005-1:2005). `PDFARevisionNumber` variables are used to configure the PDF/A Revision Number property for the generatePDFOutput and transformPDF operations of the Output service.

For information about data that can be accessed using Xpath Expressions, see Dataitems.

For information about configuring default properties, see Datatypespecific settings.

## Data items

`PDFARevisionNumber` data values stores only the string value Revision_1.

## Datatype specific settings

Property specifying the PDF/A revision number.

**Default Value**

The compliance level for a PDF/A document output. Select this value:

**Revision_1:**

PDF/A Revision 1.

# 21.84. PDFAValidationOptionSpec

A complex data type that represents the options to use for validating whether a PDF document conforms to PDF/A. This data type is used by the ValidatePDF/A operation of the DocConverter service.

For information about data that can be accessed using Xpath Expressions, see Dataitems.

For information about configuring default properties, see Datatypespecific settings.

## Data items

The data items that `PDFAValidationOptionSpec` variables contain.

### allowCertificationSignatures

A `boolean` value that specifies whether the certification signatures are permitted. The default value of `true` means that certification signatures are permitted. A value of `false` means that the certification signatures are not allowed.

### compliance

A `string` value that represents the compliance level of the PDF document. The values are `PDFA_1B`, which is the minimal level for documents scanned from paper or microfiche sources, and PDF/A-2b.

### ignoreUnusedResource

A `boolean` value that specifies whether to perform validation on resources that are not used in the PDF document. The default is `true`, which means to not perform validation on unused resources. A value of `false` means to perform validation on unused resources.

### logLevel

A `string` value that represents the log level to set. The default is `INFO`. These values are valid:

**OFF:**

Turn off logging of messages on the server.

**SEVERE:**

Log only messages on the server that indicate a serious failure.

**WARNING:**

Log only messages that indicate a potential problem.

**INFO:**

Log only informational messages.

**CONFIG:**

Log only static configuration messages.

**FINE:**

Log tracing information.

**FINER:**

Log fairly detailed tracing information.

**FINEST:**

Log highly detailed tracing information.

**NOTE:** The use of `FINE`, `FINER`, and `FINEST` will negatively affect performance on the server. Use for debugging purposes only.

**resultLevel**

A `string` value that represents the logging level to use. The default value is `PASS_FAIL`. These values are valid:

**PASS_FAIL:**

A report that contains the following information about the operation result:

- Whether the operation was successful
- Whether the PDF document is PDF/A-compliant
- Whether certified digital signatures are permitted

**SUMMARY:**

A report contains the following information about the operation result:

- Whether the operation was successful
- Whether the PDF document is PDF/A-compliant
- Whether certified digital signatures are permitted
- A summarized count of any validation errors that occurred

**DETAILED:**

A report contains the following information about the operation result:

- Whether the operation was successful
- Whether the PDF document is PDF/A-compliant
- Whether certified digital signatures are permitted
- A list of each error that occurred with specific information about each error

## Datatype specific settings

Properties for validating a PDF document.

### Compliance

The compliance level of the PDF document. The default value is `PDFA_1B`.

### Result Level

The level of results provided when the operation runs. The default value is `PASS_FAIL`.

### Allow Certification Signatures

Determines whether the certification signature is permitted. If selected, the certification signature is allowed. This option is selected by default.

### Ignore Unused Resources

Determines whether to perform validation on resources that are not used in the PDF document. If selected, validation of the unused resources is not performed. This option is selected by default.

### Job Log Level

The logging level to set. The default is `INFO`.

# 21.85. PDFAValidationResult

A complex type that is used to store the results of the ValidatePDF/A operation that the DocConverter service provides.

For information about data that can be accessed using Xpath Expressions, see Dataitems.

## Data items

Provides descriptions of each data item that `PDFAValidationResult` variables contain.

### isPDFA

A `boolean` value that indicates whether a PDF document conforms to PDF/A. A value of `true` means that the PDF document conforms to PDF/A, and `false` means that the PDF document does not conform to PDF/A.

### jobLog

A `document` value that represents a log file for a validation operation.

### validationLog

A `document` value that represents the validation results.

# 21.86. PDFDocumentVerificationInfo

A complex type that is used to store the results of the VerifyPDF Document operation of the Signature service.

For information about data that can be accessed using Xpath Expressions, see Dataitems.

## Data items

Provides descriptions of each data item that `PDFDocumentVerificationInfo` variables contain.

### description

A `string` value that describes the results of the overall validity of the document. These string values correspond to the status:

**SignedDocUnknown:**

The PDF document is signed and at least one signature requires validating.

**SignedDocInvalid:**

The PDF document is signed but at least one signature is invalid.

**SignedDocIdentityUnverifable:**

The PDF document is signed and the identity of at least one of the signers could not be verified.

**SignedDocValidAndModified:**

The PDF document is signed and all signatures are valid. In addition, unsigned changes have occurred since the PDF document was signed.

**SignedDocValid:**

The PDF document was signed and all signatures are valid.

**CertifiedDocUnknown:**

The validity of the certification for the PDF document is unknown. The certificate could not be verified.

**CertifiedDocInvalid:**

The PDF document certification is invalid.

**CertifiedDocIdentityUnverifable:**

The validity of the PDF document certification is unknown. The author of the certificate cannot be verified.

**CertifiedDocValid:**

The PDF document certification is valid.

**status**

A `PDFDocumentVerificationStatus` value that represents the status of the signature. These string values are valid:

**SignedDocUnknown:**

The status of the signed PDF document cannot be determined and is unknown.

**SignedDocInvalid:**

The status of the signed PDF document is invalid.

**SignedDocIdentityUnverifable:**

The status of the signed PDF document cannot be verified.

**SignedDocValidAndModified:**

The status of the signed PDF document is valid and modified.

**SignedDocValid:**

The status of the signed PDF document is valid.

**CertifiedDocUnknown:**

The status of the certified PDF document cannot be determined and is unknown.

**CertifiedDocInvalid:**

The status of the certified PDF document is invalid.

**CertifiedDocIdentityUnverifable:**

The status of the certified PDF document cannot be verified.

**CertifiedDocValid:**

The status of the certified PDF document is valid.

**verificationInfos**

A `list` of `PDFSignatureVerificationInfo` values that represent information about each digital signature that is verified in the PDF document.

# 21.87. PDFFormRenderSpec

A complex data type that represents PDF rendering options. `PDFFormRenderSpec` variables are used to configure the PDF Form Render Options property in the renderFormGuide, renderPDFForm operations of the Forms service.

For information about data that can be accessed using Xpath Expressions, see Dataitems.

For information about configuring default properties, see Datatypespecific settings.

## Data items

Provides descriptions of the data items that `PDFFormRenderSpec` variables contain.

### acrobatVersion

An `acrobatVersion` value that represents the Acrobat version that is required to view the PDF form that the Forms service renders.

### cacheEnabled

A `boolean` value that indicates whether the Forms service caches a PDF form to improve performance. When caching is used, each form is cached after it is generated for the first time. On a subsequent render, if the cached form is newer than the form design's timestamp, the form is retrieved from the cache. A value of `true` indicates that caching is used and `false` indicates caching is not used.

### charset

A `string` value that represents the character set used to encode the output byte stream. The Forms service supports character encoding values defined by the`java.nio.charset` package for HTML transformations. For a complete list of character sets, see http://java.sun.com/j2se/1.5.0/docs/guide/intl/encoding.doc.html

### clientCache

A `boolean` value that determines whether the form is cached in the client web browser cache. Only forms that are rendered as interactive PDF forms can be stored in the client web browser cache.

When client caching is used and subsequent requests for the PDF form are made, a timestamp located in the cached PDF form is compared with the timestamp of the PDF form that is generated by the Forms service and stored in the server cache. If they are the same, the PDF form is retrieved from the client cache. This results in reduced bandwidth usage and improves performance because the Forms service does not have to redeliver the same content to the client web browser.

A value of `true` indicates that client caching is used and `false` indicates client caching is not used.

### debugEnabled

A `boolean` value that determines whether debug-level logging is performed. Debug-level logging provides more information in the J2EE application server's log file. A value of `true` indicates that debug-level logging is performed, and `false` indicates that the default level of logging is performed.

### formModel

A `FormModel` value that represents where the form processing is performed.

### generateServerAppearance

A `boolean` value that determines whether the appearance, such as layout of PDF pages and other graphical elements, for the PDF form are generated on the server. When you generate the appearance

on the server, the form is rendered on the server and merged with data. A value of `true` specifies to generate the appearance on the server.

**linearizePDF**

A `boolean` value that indicates whether the Forms service produces a linearized PDF form, which is a form that is optimized for web applications. A linearized PDF document enables incremental access in a network environment. For example, a linearized PDF document can be displayed in a web browser before the entire PDF document is downloaded. A value of `true` indicates the PDF is linearized and `false` indicates it is not linearized.

**locale**

A `string` value that represents the locale to use.

**PDFVersion**

A `string` value that represents the PDF version of the PDF form that is rendered. These string values are valid:

**PDFVersion_1_5:**

**PDFVersion_1_6:**

Represents PDF version 1.6.

**PDFVersion_7:**

Represents PDF version 1.7.

**renderAtClient**

A `RenderAtClient` value that determines whether PDF content is rendered on the client application using Acrobat 7.0 or later, or on the server.

**seedPDF**

A `string` value that represents the location of a shell PDF (seed PDF) to use for optimizing the delivery of transformed PDF documents. The shell PDF document specifies a customized PDF document (contains only fonts) that is appended with a form design and data. The form is rendered by Acrobat 7.0 or later and applies to PDF form transformations.

**standAlone**

A `boolean` value that specifies whether the form is rendered without state information. This value is useful when the rendering of an interactive form occurs on the server and the form contains JavaScript code to execute.

A value of `true` indicates that the JavaScript code that the form contains runs on the client with no interaction with the server. A value of `false` indicates that state information is used to render an interactive

form to an end user who then enters information into the form and submits the form back to the Forms service. The Forms service then performs a calculation operation and renders the form back to the user with the results displayed in the form.

### taggedPDF

A `boolean` value that determines whether the Forms service produces a tagged PDF form. A value of `true` indicates a tagged PDF form is produced and a value of `false` indicates the PDF form is not tagged. A tagged PDF form defines a set of standard structure types and attributes that support the extraction of page content and reuse for other purposes. It is intended for tools that perform simple extraction, automatic redirection of flows, processing of text for searching or indexing, conversion to other common file formats, and making content accessible to users with visual impairment.

### XCIURI

A `string` value that represents the URI of the XCI file to use for rendering. If the root of the URI is not specified, the file is assumed to reside in the EAR file.

### XMLData

A `boolean` value that specifies to the Forms service to create XML data after the operation executes. A value of `True` specifies to produce the XML form's data based on its current processing state and a value of `False` specifies not to produce XML data.

## Datatype specific settings

Properties for specifying the default values of `PDFFormRenderSpec` variable data items.

### Character Set

Sets the character set used to encode the output byte stream. The default value is <Use Server Default>. Select the character set to use or one of these values.

**<Use Server Default>:**

Use the Character Set setting configured on AEM Forms Server. The Character Set setting is configured using AEM Forms - Forms Generator in administration console. (See AEM Forms administration help.)

**<Use Custom Value>:**

Use a character set that is not available in the list. After selecting this value, in the text box beside the list, type the canonical name (Java.nio API) of the encoding set to use. For a complete list of character sets, see http://java.sun.com/j2se/1.5.0/docs/guide/intl/encoding.doc.html.

### Locale

Sets the language used to send validation messages to client devices, such as web browsers, when an HTML form is rendered. The default value is <Use Server Default>. Select either a language from the list or one of these values.

**<Use Server Default>:**

Use the Locale setting configured on AEM Forms Server. The Locale setting is configured using Forms in administration console. (See AEM Forms administration help.)

**<Use Custom Value>:**

Use a locale that is not available in the list. After selecting this value, in the text box beside the list, type the Locale ID of the locale code to use. For a complete list of supported locale codes, see http://java.sun.com/j2se/1.5.0/docs/guide/intl/locale.doc.html.

### Cache Form On Server

Sets whether the form design is cached on the server. Caching forms on the server improves performance for forms that are rendered on the server. When no value is selected, the Form Cache Control settings are used. The Form Cache Control Settings are configured using Forms in administration console. (See AEM Forms administration help.) Select one of these values:

**False:**

The form is not cached on the server.

**True:**

The form is cached on the server.

### Acrobat Version

Sets the minimum version of Acrobat and Adobe Reader version required to open the rendered PDF form and the PDF Version used to create the PDF form. The default value is <Use Form Template Default>. Select one of these values:

**<Use Form Template Default>:**

The minimum version of Acrobat or Adobe Reader version is specified by the Target Version setting in the form design. The PDF Version that is used is determined by form design settings.

**Acrobat and Adobe Reader 6 or later:**

**Acrobat and Adobe Reader 7.0 or later:**

Acrobat or Adobe Reader 7.0 and later can open the PDF form. PDF Version 1.6 is used.

**Acrobat and Adobe Reader 7.0.5 or later:**

Acrobat or Adobe Reader 7.0.5 and later can open the PDF form. PDF Version 1.65 is used.

**Acrobat and Adobe Reader 8 or later:**

Acrobat or Adobe Reader 8 and later can open the PDF form. PDF Version 1.7 is used.

**Acrobat and Adobe Reader 8.1 or later:**

Acrobat or Adobe Reader 8.1 and later can open the PDF form. PDF Version 1.7-ADBE-1 is used.

**Acrobat and Adobe Reader 9 or later:**

Acrobat or Adobe Reader 9 and later can open the PDF form. PDF Version 1.7-ADBE-3 is used.

**Populate XML Data**

Sets whether the XML data is produced from the form design based on the its current processing state. The default value is False. Select one of these values:

**False:**

Do not produce the XML data.

**True:**

Produce the XML data.

**Tagged PDF**

Sets whether to create a tagged Adobe PDF form. A tagged PDF form defines a set of standard structure types and attributes that support the extraction of page content and reuse for other purposes. It is intended for tools that perform simple extraction, automatic redirection of flows, processing of text for searching or indexing, conversion to other common file formats, and making content accessible to users with visual impairment. It is intended for use by tools that perform the following types of operations:

- Simple extraction of text and graphics for pasting into other applications.
- Automatic reflow of text and associated graphics to fit a page of a different size than was assumed for the original layout.
- Processing text for such purposes as searching, indexing, and spell-checking.
- Conversion to other common file formats (such as HTML, XML, and RTF) with document structure and basic styling information preserved.
- Making content accessible by screen reader software.

The default value is True. Select one of these values:

**False:**

Do not render a tagged PDF form.

**True:**

Render a tagged PDF form.

**Linearized PDF**

Sets whether to render a linearized PDF form. A linearized PDF form is organized so that it supports incremental access in a network environment. For example, a linearized PDF can be displayed in a web browser before the entire PDF document is downloaded. The default value is False. Select one of these values:

**False:**

Do not render a linearized PDF form. It is best to use this option for non-web applications.

**True:**

Render a linearized PDF form. It is best to use this option for optimized web applications.

**Seed PDF**

Sets the name of the initial PDF form that is used in a PDF form to optimize delivery. The seed PDF form specifies a customized PDF form (contains only fonts) that is appended with a form design and data. The form design is rendered by Acrobat 7.0 or later and applies to the PDFForm and PDFMerge transformations.

**Render At Client**

Sets whether to enable the delivery of PDF content by using the client-side rendering capability for Acrobat 7.0 or Adobe Reader and later. Client-side rendering improves the performance of Forms and only applies to PDF, PDFForm, or PDFMerge transformations. The default value is <Use Server Default>. Select one of these values:

**<Use Server Default>:**

Use the Render At setting to specify the version of Acrobat and Adobe Reader required to open a PDF Forms. The Render At setting is configured in administration console. (See AEM Forms administration help.)

**<Use Form Template Default>:**

The Forms service determines the form rendition based on the setting in the form design.

**Yes:**

**No:**

A static PDF form is generated. No rendering on the client occurs.

**Stand Alone Rendition**

Sets whether the form can be rendered without state information. State information is used for rendering interactive forms that require user interaction with the server for submissions.The default value is False. Select one of these values:

**False:**

The form is rendered without state information and without embedded JavaScript that runs on the client web browser. This means that the Forms service is used to render the form back to the user when performing actions such as calculations. The results of the calculation are displayed in the rendered form. Because of the required interaction with the Forms service, the form cannot be used offline.

**True:**

The form is rendered with state information and with embedded JavaScript. The JavaScript code runs on the client with no interaction with the server. In addition, the form can be used offline.

### Form Model

Sets the location where scripts embedded in the form are to be executed. The default value is <Use Form Template Default>. Select one of these values:

**<Use Form Template Default>:**

The Forms service checks the form design to determine whether to render on the client or server.

**<Client Side>:**

The form is rendered on the client. Do not use scripts that run on the server. When this value is selected, scripts that run on the server generate a warning on AEM Forms Server.

**<Server Side>:**

The form is rendered on the server.

**<Both Server and Client side>:**

The form is rendered on both the server and the client.

### XCI URI

Type a value to specify the URI location of the XCI file to use for rendering. If the root is not specified, the file is assumed to reside in the location where the EAR files are deployed.

### Client Cache

Sets whether the rendered PDF form is cached on the client web browser. Only forms that are rendered as interactive PDF forms can be stored in the client web browser cache.

When client caching is used, the timestamps of the cached PDF form is compared with the timestamp of the PDF form generated on the server. If the timestamps are the same, the PDF form is retrieved from the client cache. When the compared timestamps are different, the server redelivers the PDF form. Using the cache on the client results in reduced bandwidth usage and improves performance. Performance improves because the Forms service does not have to redeliver PDF form to the client application. The default value is False. Select one of these values:

**False:**

Do not cache the form on the client.

**True:**

Cache the form on the client.

### Generate Server Appearance

Sets whether the appearance for the PDF form is generated on the server. Appearances include the layout of fields and graphical elements in the PDF form. When you generate the appearance on the server, the form is rendered on the server and merged with data. Generate the appearance on the server to use the rendered PDF form in a subsequent operation.

**False:**

(Default) Do not generate the appearance on the server.

**True:**

Generate the appearance of the rendered PDF form on the server.

# 21.88. PDFLegalWarnings

A complex data type that represents legal warnings in a PDF document that has digital signatures applied to it. `PDFLegalWarning` variables are used to configure the GetLegal Attestation operation of the Signature service. It contains information about legal warnings located in the PDF document.

For information about data that can be accessed using Xpath Expressions, see Dataitems.

## Data items

The data items that `PDFLegalWarnings` variables contain.

### alternateImagesCount

An `int` value that represents the number of alternate images found in the PDF document.

### annotationsCount

An `int` value that represents the number of annotations located in the PDF document.

### catalogHasAACount

An `int` value that represents the number of additional action values. These values define the actions taken in response to various trigger events affecting the PDF document. An example is an action that is performed when the page that contains the annotations is opened. For a full list of these trigger events, see PDF Utilities.

### catalogHasOpenAction

An `int` value that represents the number of open action values in the PDF document. An *open action* specifies the destination displayed or the action performed when a PDF document is opened.

### devDepGS_SMaskCount

An `int` value that represents the SMask value. A *soft-mask (SMask)* is a value that specifies the mask shape or opacity used in the transparent image.

### devDepGSBGCount

An `int` value that represents the number of references to the graphics state parameter BG (black-generation) found in the PDF document.

### devDepGSFLCount

An `int` value that represents the number of references to the graphics state parameter FL (flatness tolerance) found in the PDF document.

### devDepGSHTCount

An `int` value that represents the number of references to the graphics state parameter HT (halftone) found in the PDF document.

### devDepGSOPCount

An `int` value that represents the number of references to the graphics state parameter OP (overprint) found in the PDF document.

### devDepGSTRCount

An `int` value that represents the number of references to the graphics state parameter TR (transfer) found in the PDF document.

### devDepGSUCRCount

An `int` value that represents the number of references to the graphics state parameter UCR (undercolor removal) found in the PDF document.

### docHasCryptFilter

An `int` value that represents the number of crypt filters located in the PDF document. A *crypt filter* lets the security handler determine which algorithms are used to decrypt data.

### docHasNonSigField

An `int` value that represents the number of non-signature fields located in the PDF document.

### docHasPresentation

An `int` value that represents the number of presentations included in the document. A PDF document can contain different presentations that specify the alternative ways in which the document can be viewed.

### docHasPSXObj

An `int` value that represents the number of PostScript XObjects located in the PDF document. A *PostScript XObject* includes PostScript language fragments that are used when printing to a PostScript output device.

### docHasRequirementHandler

An `int` value that represents the number of requirement handlers located in the PDF document. A *requirement handler* is a program that verifies certain requirements are satisfied. For information about requirement handlers, see PDF Utilities.

### docHasXFA

An `int` value that represents the number of XFA entries located in the PDF document. An Adobe XML Form Architecture (XFA) entry can be either a stream that contains the entire XFA resource or an array that specifies individual packets that together make up the XFA resource.

### dynamicSigAPCount

An `int` value that represents the number of AP entries related to the signature field. The *appearance dictionary (AP)* entry defines the visual appearance of the signature field.

### externalOPIDictsCount

An `int` value that represents the number of OPI dictionaries found in the PDF document. The *Open Prepress Interface (OPI)* is a mechanism for creating low-resolution placeholders (proxies) for high-resolution images.

### externalStreamsCount

An `int` value that represents the number of external streams found in the PDF document.

### futurePDFVersionCount

An `int` value, for internal use only.

### goTo3DViewActionCount

An `int` value that represents the number of go-to-3D-view actions found in the PDF document. A *go-to-3D-view* action identifies a 3D annotation and specifies a view for the annotation to use. For information about go-to-3Dview actions, see PDF Utilities.

### goToEHasF

An `int` value that represents the number of *Go-to-embedded (*GoToE) actions with a specified target found in the PDF document. A *GoToE* action moves the focus to a destination in an embedded file. For information about GoToE actions, see PDF Utilities.

### hideActions

An `int` value that represents the number of hide actions found in the PDF document. A *hide* action hides or shows one or more annotations on the screen by setting or clearing their hidden flags. This type of action can be combined with appearance streams and trigger events to display pop-up window help information on the screen.

### hideAnnotationActions

An `int` value that represents the number of hide actions found in the PDF document. A *hide* action hides or shows one or more annotations on the screen by setting or clearing their hidden flags.

### importDataActions

An `int` value that represents the number of import-data actions found in the PDF document. An *import-data* action imports Forms Data Format (FDF) data into the document's interactive form from a specified file.

### invalidEOF

An `int` value that represents the number of invalid end-of-file markers in the PDF document. Acrobat requires that an end-of-file marker appear somewhere within the last 1024 bytes of the file.

### invalidFileHeader

An `int` value that specifies whether the first line in the PDF document is a valid header. A valid header identifies the version of the PDF specification to which the file conforms.

### javaScriptActions

An `int` value that represents the number of JavaScript actions found in the PDF document. A *JavaScript* action causes compilation and execution of scripts by the JavaScript interpreter.

### launchActions

An `int` value that represents the number of launch actions found in the PDF document. A *launch* action opens an application, opens a document, or prints a document.

### legalAttestationString

A `string` value that contains a legal attestation created by the author of the PDF document. It explains the presence of any other entries in the dictionary or the presence of any other content affecting the legal integrity of the document.

### malformedContentStm

An `int` value that represents the number of malformed content stream occurrences found in the PDF document. A *content stream* is a PDF stream object whose data consists of a sequence of instructions that describe the graphical elements to be painted on a page.

### movieActions

An `int` value that represents the number of movie actions found in the PDF document.

### namedAction

An `int` value that represents the number of named actions found in the PDF document. A *named* action executes an action predefined by the viewer application. The standard named actions supported by the PDF viewer are: NextPage, PrevPage, FirstPage, and LastPage.

### nonEmbeddedFontsCount

An `int` value that represents the number of non-embedded fonts found in the PDF document.

### optionalContentPresent

A `boolean` value that specifies whether optional content was found in the PDF document. A value of `true` indicates that the PDF document contains optional content, and a value of `false` indicates that the PDF document does not contain optional content.

### pageHasAA

An `int` value that represents the number of additional-action entries found in the PDF document. These entries define the behavior of the page in response to various trigger events. For a full list of these trigger events, see PDF Utilities.

### refXobjects

An `int` value that represents the number of reference XObjects located in the PDF document. A *reference XObject* enables a PDF document to import content from another PDF document.

### renditionActionCount

An `int` value that represents the number of rendition actions located in the PDF document. A *rendition* action controls the playing of multimedia content.

### setOCGStateActions

An `int` value that represents the number of set-OCG-state actions located in the PDF document. A *set-OCG-state* action sets the state of one or more optional content groups. For information about the set-OCG-state actions, see PDF Utilities.

### setStateActions

An `int` value that represents the number of set-OCG-state actions in the PDF document. These actions set the state of one or more optional content groups. For more information about the set-OCG-state actions, see PDF Utilities.

**sigFieldHasAA**

An `int` value that represents the number of additional action entries that are related to the signature field, found in the PDF document. These entries define the behavior of the field in response to various trigger events. For a full list of these trigger events, see PDF Utilities.

**sigFieldHasAction**

An `int` value that represents the number of signature field actions located in the PDF document. A *signature field* action indicates that the set of fields are locked when the signature field contains a signature.

**soundActions**

An `int` value that represents the number of sound actions found in the PDF document.

**trueTypeFontsCount**

An `int` value that represents the number of TrueType fonts found in the PDF document.

**unknownNamedAction**

An `int` value that represents the number of unknown named actions found in the PDF document. A *named* action executes an action predefined by the viewer application. The standard named *actions* supported by the PDF viewer are NextPage, PrevPage, FirstPage, and LastPage.

**unknownPDFContent**

An `int` value that represents the number of unknown data items found in the PDF document.

**uriActions**

An `int` value that represents the number of URI actions found in the PDF document. A *uniform resource identifier (URI*) is a string that identifies a resource on the Internet.

**XObjHasInterpolate**

An `int` value that represents the number of interpolate entries found in the PDF document. When the resolution of a source image is lower than the resolution of the output device, each source sample covers many device pixels. As a result, images can appear jaggy or blocky. These visual artifacts can be reduced by applying an image interpolation algorithm during rendering. Instead of painting all pixels covered by a source sample with the same color, image interpolation attempts to produce a smooth transition between adjacent sample values. Image interpolation is enabled by setting the Interpolate entry in the image dictionary to true.

## 21.89. PDFOutputOptionsSpec

A complex data type that represents options and settings to generate a PDF document. `PDFOutputOptionsSpec` variables are used to configure the PDF Output Options property in the generatePDFOutput(deprecated) operation of the Output service.

For information about data that can be accessed using Xpath Expressions, see Dataitems.

For information about configuring default properties, see Datatypespecific settings.

## Data items

The data items that `PDFOutputOptionsSpec` variables contain.

### charset

A `string` value that represents the character set to use for the PDF output. For a list of character sets, see http://java.sun.com/j2se/1.5.0/docs/guide/intl/encoding.doc.html.

### fileURI

A `string` value that represents the URI of a file to which the Output service writes the output.

### generateManyFiles

A `boolean` value that specifies whether the Output service creates single output or multiple outputs. A value of `true` indicates that multiple outputs are generated and `false` indicates single output is generated.

### lazyloading

A `boolean` value that specifies whether incremental (lazy) loading is used when processing multi-record data sets. When set to the a value of `true`, multi-record data sets are loaded and merged one record of data at a time. Processing one record at a time helps to avoid running out of memory. The use of incremental loading limits XLST options specified in the XCI file because transformations can only be applied to only one record.

### locale

A `string` value that indicates the locale that the Output service uses to render the PDF document. For a list of supported locale codes, see http://java.sun.com/j2se/1.5.0/docs/guide/intl/locale.doc.html.

### lookAhead

An `int` value that represents the number of bytes used for search rules. This value is the number of bytes to use from the beginning of the input data file to scan for the pattern strings. The default value is `500`.

### lpdURI

A `string` value that represents the URI of the Line Printer Daemon (LPD) to use when the network has an LP daemon running.

### metaDataSpecFile

A `boolean` value that indicates whether metadata is generated. A value of `true` indicates that metadata is generated and `false` indicates that metadata is not generated.

### printerQueueName

A `string` value that represents the name of the printer queue that is used with the Line Printer Daemon (LPD) URI.

### printerURI

A `string` value that represents the URI of the destination for the print output. The Output service sends a print stream to the printer specified by the URI for this data item (for example, `\\PrintServer\Print1`).

### recoredIDField

A `string` value that represents the identifier value for the batch record.

### recordLevel

An `int` value that represents the element level (located within the input data file) that contains data records. The root element is level 1. This value cannot be used with the recordName value.

### recordLevelMetaData

A `boolean` value that specifies whether metadata is generated. A value of `true` indicates that metadata is generated and `false` indicates metadata is not generated.

### recordName

A `string` value that represents the element name that identifies the beginning of a record in the input data file. Any value indicates that the input data contains record batches. This value cannot be used with the recordLevel value.

### rules

A `list` of `string` values that contains `string` values that represent search rules that scan the input data file for a pattern and associates the data with a specific form design.

### serverPrintSpec

A `string` value that represents the print specification name. A print specification contains information that the Output service requires during each invocation request. When you configure a print specification, the Output service sends the output to a network printer, an email recipient as a file attachment, or a file such as a PostScript file.

### XCIURI

A `string` value that specifies the XCI file to use when generating PDF document output. An XCI file is a configuration file that the Output service uses. By default, the Output service uses an XCI file named pa.xci. Specify a value for this data item to use a different file.

## Datatype specific settings

Properties for configuring the rendering options for PDF document.

### General

*XCI URI*

Sets the URI of where the XCI file is located.

*Character Set*

Sets the character set that is used to encode the rendered form. Select the character set to use or one of these values.

**<Use Server Default>:**

(Default) The Character Set setting that is configured on AEM forms Server. This setting is configured using administration console. (See AEM Forms administration help.)

**<Use Custom Value>:**

Use a character set that is not in the list. After selecting this value, in the box beside the list, type the canonical name (Java.nio API) of the encoding set that is not in the list. For a list of character sets, see http://java.sun.com/j2se/1.5.0/docs/guide/intl/encoding.doc.html.

*Locale*

Sets the language used for generating the PDF document. Select a language to use from the list or one of these values.

**<Use Server Default>:**

(Default) Use the Locale setting configured on AEM Forms Server. The Locale setting is configured using administration console. (See AEM Forms administration help.)

**<Use Custom Value>:**

Use a locale that is not in the list. After selecting this value, in the box beside the list, type the Locale ID of the locale code to use. For a complete list of supported locale codes, see http://java.sun.com/j2se/1.5.0/docs/guide/intl/locale.doc.html.

### Batch

*Record Name*

Sets the name of the element that identifies the beginning of a batch of records.

*Record Level*

Sets the XML element level that contains the record data. The default is 1, which represents the first level of the record that contains data. The first level is typically below the element specified by the Record Name property.

*Generate Multiple Streams*

Sets whether the operation creates a single or multiple outputs. Select one of these values:

**True:**

Create multiple outputs.

**False:**

(Default) Create a single output.

*Enable Lazy Loading*

Sets whether incremental (lazy) loading is used when processing multi-record data sets. Select one of these values:

**True:**

Multi-record data sets are loaded and merged one record of data at a time. Processing one record at a time helps to avoid running out of memory, but limits the use of XSLT in the XCI file. The use of incremental loading limits XLST options specified in the XCI file because transformations can only be applied to only one record.

**False:**

All records are loaded and merged at one time because the entire data file is loaded.

**Rules**

*Pattern Match Size*

Sets the number of bytes to use from the beginning of the input data file to scan for the pattern strings. The default is 500.

*Pattern Matching Rules*

Sets rules for scanning the input data file for a pattern and associates the data with a specific form design.

**Add A List Entry:**

Adds a new rule. After you click this button, a new entry is created in the list. In the Pattern field for the new entry, type a pattern to search for. In the Form field, type the name of the form design for the matching pattern. All the form designs that you specify must be available at the location specified by the Content Root property in this operation.

**Delete A Selected List Entry:**

Removes the selected rule from the list.

For information about working with search rules, see Designer Help.

**Destination**

*Output Location URI*

Sets the URI and name of the output file to save.

*Printer Name*

Sets the name of the printer for sending the output for printing.

*LPD URI*

Sets the URI of the Line Printer Daemon (LPD) to use.

*LPD Printer Name*

Sets the name of the printer on the specified Line Printer Daemon (LPD) to use.

**MetaData**

*Meta Data Spec File*

Sets the URI of the metadata spec file to use. A metadata spec file is used to generate metadata from the provided data file.

*Record ID XPath*

Sets the root level node to use for XPath expressions. The root level node specifies the starting point for XPath expressions.

*Generate Record Level Meta Data*

Sets whether to generate a metadata file for each record. Select one of these values:

**True:**

Generate a metadata file for each record that is processed.

**False:**

(Default) Generate one metadata file for all records that are processed.

# 21.90. PDFPropertiesOptionSpec

A complex data type that represents a query for PDF documents to determine the service that must be used. For example, to create a PostScript file from a PDF document, you must determine whether the document is a PDF document or an XFA document. If the document is a PDF document, it can be processed by the Convert PDF service. If the document is an XFA document, it can be sent to the Output service. This data type is used in the GetPDF Properties operation of the PDF Utilities service.

For information about data that can be accessed using Xpath Expressions, see Dataitems.

## Data items

The data items that `PDFPropertiesOptionSpec` variables contain.

### hasAttachments

A `boolean` value that indicates whether to check the document for file attachments. A value of `true` means to check whether the document has attachments and `false` means not to check for attachments.

### hasComments

A `boolean` value that indicates whether to check the document for comments. A value of `true` means to check whether the document contains comments and `false` means not to check for comments.

### isAcroForm

A `boolean` value that indicates whether to check if the document is an Acrobat form (a form that is created in Acrobat). A value of `true` means to check if the form is created in Acrobat and `false` means not to check if the document is an Acrobat form.

### isFillableForm

A `boolean` value that indicates whether to check if the document contains fillable fields. A value of `true` means to check if the document contains fillable fields and `false` means not to check for fillable fields.

### isPDFDocument

A `boolean` value that indicate whether to check if you can perform PDF document operations on this document. A value of `true` means to check if the document allows performing PDF operations and `false` means not to check if you can perform PDF operations.

### isPDFPackage

A `boolean` value that indicates whether to check if the document is a PDF package. A *PDF package* is a portable collection that uses metadata, including user-defined metadata, to manage a collection of document-level file attachments. The root PDF, to which all the files are attached, is less important than the attachments and is referred to as a cover sheet within Acrobat. A value of `true` means to check if this is a PDF package and `false` means not to check if this is a PDF package.

### isXFADocument

A `boolean` value that indicates whether to check if the document is an XFA document. A value of `true` means to check if this is an XFA document and `false` means not to check if this is an XFA document.

### queryFormType

A `boolean` value that indicates whether to check the form type of the document. A value of `true` means to check the form type of the document and `false` means not to check the form type.

### queryPDFVersion

A `boolean` value that indicates whether to retrieve the PDF version of the document. A value of `true` means to retrieve the PDF version and `false` means not to retrieve the PDF version of the document.

### queryRequiredAcrobatVersion

A `boolean` value that indicates whether to retrieve the version of Acrobat or Adobe Reader required to correctly view the document. A value of `true` means to retrieve the Acrobat version and `false` means not to retrieve the Acrobat version.

### queryXFAVersion

A `boolean` value that indicates whether to retrieve the XFA version of the document. A value of `true` means to retrieve the XFA version of the document and `false` means not to retrieve the XFA version.

# 21.91. PDFPropertiesResult

A complex data type that represents the results of the GetPDF Properties operation that the PDF Utilities service provides. This variable contains various document information, such as the version, whether it contains comments, or whether it contains attachments.

For information about data that can be accessed using Xpath Expressions, see Dataitems.

## Data items

The data items that `PDFPropertiesResult` variables contains.

### formType

A `string` value that represents the form type of the PDF document. These values are valid:

- NotAForm
- Acroform
- Static-XFA
- Dynamic-XFA
- XFAForeground

### hasAttachments

A `boolean` value that determines whether the document has attachments. A value of `true` indicates that the document contains attachments and `false` indicates that there is no attachments in this document.

### hasComments

A `boolean` value that indicates whether the document contains comments. A value of `true` indicates that there are comments in the document and `false` indicates that there are no comments in this document.

Comments can be one of the following types:

- Notes (Text)
- DrawingMarkups (Line, PolyLine, Square, Circle, Polygon, and Ink)
- TextEditingMarkups (Highlight, Underline, Squiggly, StrikeOut, Caret, and FreeText)
- Stamps (Stamp)
- Attachments (FileAttachment and Sound)

### isAcroForm

A `boolean` value that indicates whether the document is an Acrobat form that does not contain an XFA stream. A value of `true` indicates that the document is an Acrobat form and `false` indicates that it is not an Acrobat form.

### isFillableForm

A `boolean` value that indicates whether the document contains fillable form fields. A value of `true` indicates that there are fillable fields in the document and `false` indicates that there is no fillable fields in the document.

### isPDFDocument

A `boolean` value that indicates whether you can perform PDF document operations on this document. A value of `true` indicates that the document allows for PDF operations and `false` indicates that you cannot perform PDF operations in this document.

**NOTE:** If the document contains an XFA stream, but is an XFA Foreground stream, it is still a PDF document. An XFA Foreground stream is a PDF document that displays an XFA form as a layer on applicable pages.

### isPDFPackage

A `boolean` value that indicates whether the document is a PDF package. A *PDF package* is a portable collection that uses metadata, including user-defined metadata, to manage a collection of document-level file attachments. The root PDF, to which all the files are attached, is less important than the attachments and is referred to as a cover sheet within Acrobat. A value of `true` indicates that the document is a PDF package and `false` indicates that the document is not a PDF package.

### isXFADocument

A `boolean` value that indicates whether the document contains an XFA stream. A value of `true` indicates that the document contains an XFA stream and `false` indicates that the document does not contain an XFA stream.

### locked

A `boolean` value that indicates whether the document is locked. A value of `true` indicates that the document is locked and `false` indicates that this document is not locked.

### PDFVersion

A `string` value that contains the document's PDF version. This information is located in the first line of the PDF document (header), and it specifies the version of the PDF specification to which the file conforms. For example, if the file conforms to PDF version 1.5, the header contains the string: `%PDF-1.5`. The valid values are: `PDF-1.5`, `PDF-1.6`, and `PDF-1.7`.

### queryExceptions

A `map` of `string` values that contains exceptions that are thrown when the document property values are retrieved.

### requiredAcrobatVersion

A `string` value that contains the Acrobat or Adobe Reader version required to view the PDF document.

### XFAVersion

A `string` value that contains the XFA version of the PDF document.

# 21.92. PDFSeedValueOptionSpec

A complex data type used by the AddInvisible Signature Field and AddVisible Signature Field operations of the Signature service. It represents the seed value dictionary that is associated with a signature field. A seed value dictionary contains entries that constrain information that is used when the signature is applied.

For information about data that can be accessed using Xpath Expressions, see Dataitems.

For information about configuring default properties, see Datatypespecific settings.

## Data items

The data items that `PDFSeedValueOptionSpec` variables contain.

### addRevInfo

A `boolean` value that specifies whether revocation checking is performed. A value of `true` means that the viewer application must perform the following revocation tasks when signing a signature field:

- Perform revocation checking of the certificate used to sign. The checking also includes the corresponding issuing certificates.
- Include the revocation information within the signature value.

If the `subFilterEx` value is either `adbe_pkcs7_detached` or `adbe_pkcs7_sha1`, this value must be set as `true`. If the `subFilterEx` value is `x509_rsa_sha1`, this value must be omitted or specified as `false`, or the signature process fails.

The default value of `false` means that the revocation checking is not performed.

### certificateSeedValueOptions

A `CertificateSeedValueOptionSpec` value that represents a certificate seed value dictionary. A certificate seed value dictionary provides constraining information that is used at the time the signature is applied.

`CertificateSeeValueOptionSpec` values contain the following data items:

#### flags

An `int` value that represents the flags associated with this certificate seed value. A value of 1 means that a signer is required to use only the specified values for the entry. A value of `0` means that other values are permissible. The default value is `0`.

#### issuers

A `list` of `byte` values that contains byte-encoded X.509v3 certificates of acceptable issuers. If the signer's certificate is in the chain of the listed issuers, the certificate is considered acceptable for signing.

#### keyUsage

A `string` value that specifies an acceptable key-usage extension that must be present in the signing certificate. Multiple strings are used to specify a range of acceptable extensions.

Each character in the string represents a key-usage type, where the order of the characters indicates the key-usage extension it represents. The first through ninth characters in the string represent the following key-usage extensions:

**1**: digitalSignature

**2**: non-Repudiation

**3**: keyEncipherment

**4**: dataEncipherment

**5**: keyAgreement

**6**: keyCertSign

**7**: crlSign

**8**: encipherOnly

**9**: decipherOnly

Any additional characters are ignored. Any missing characters or characters that are not one of the above values is set to X.

Each string is composed of the following characters:

**0**: Corresponding key-usage must not be set.

**1**: Corresponding key-usage must be set.

**X**: The state of the corresponding key-usage does not matter.

**NOTE:** For example, the `string` values `'1'` and `'1XXXXXXXX'` represent settings where the key-usage type digitalSignature must be set, and the state of the other key-usage types does not matter.

**oids**

A `list` of `string` values, where each string contains object identifiers (OIDs) of the certificate policies that must be present in the signing certificate.

**subjectDN**

A `list` of `string` values, where each string contains key value pairs that specify the subject Distinguished Name (DN), must be present within the certificate to be acceptable for signing. The certificate must at least contain all the attributes specified in the dictionary, but can also contain other attributes.

**subjects**

An `list` of `byte` values that contains byte-encoded X.509v3 certificates that are acceptable for signing.

**url**

A `string` value that specifies the URL that can be used to enroll for a new credential if a matching credential is not found.

**urlType**

A `string` value that contains the name indicating the usage of the URL entry. There are standard and implementation-specific uses for the URL.

The value `Browser` represents a valid standard usage. It specifies that the URL references the content to be displayed in a web browser to allow enrolling for a new credential.

The value `ASSP` represents a valid implementation-specific usage, defined for use by Adobe Systems. It specifies that the URL references a signature web service used for server-based signing.

**NOTE:** Third parties can extend the use of this item with their own values. These values must conform to the guidelines described in Appendix E of the PDF Utilities.

### digestMethod

A `HashAlgorithm` value that specifies the names of the acceptable digest algorithms to use while signing.

digestMethod can be one of these string values:

**SHA1:**

The Secure Hash Algorithm has a 160-bit hash value.

**SHA256:**

The Secure Hash Algorithm has a 256-bit hash value.

**SHA384:**

The Secure Hash Algorithm has a 384-bit hash value.

**SHA512:**

The Secure Hash Algorithm has a 512-bit hash value.

**RIPEMD160:**

The RACE Integrity Primitives Evaluation Message Digest has a 160-bit message digest algorithm.

**NOTE:** This data item is applicable only if the digital credential signing contains RSA public/private keys. If the digital credential contains DSA public/private key, the digest algorithm is always SHA1.

### filter

Deprecated, use filterEX

### filterEX

A `PDFFilterType` value that represents a signature handler that is used to sign the signature field. The only valid value is `Adobe_PPKLite`.

### flags

An `int` value that represents the flag associated with the seed value. These values are valid:

**1 (Filter):**

The signature handler used to sign the signature field.

**2 (SubFilter):**

An array of names indicating acceptable encodings to use when signing.

**3 (V):**

The minimum required version number of the signature handler used to sign the signature field.

**4 (Reasons):**

An array of strings specifying possible reasons for signing a document.

**5 (PDFLegalWarnings):**

An array of strings specifying possible legal attestations.

**6 (AddRevInfo):**

Revision information.

**7 (DigestMethod):**

A name identifying the algorithm used when computing the digest.

The default value of `0` means that the associated entry is an optional constraint.

**legalAttestations**

A `string` value that represents a legal attestation associated with the seed value.

**mdpValue**

An `MDPPermissions` value that specifies the changes that can be done on a PDF document without invalidating the signature after the legal attestations are provided.

These string values are valid:

**NonAuthorSignature:**

The signature is an ordinary signature.

**NoChanges:**

No changes to the document are allowed. Any change invalidates the signature.

**FormChanges:**

Permitted changes include filling in form, instantiating page templates, and signing the form.

**AnnotationFormChanges:**

In addition to FormChanges, other permitted changes include annotation creation, deletion, and modification.

**reasons**

A `string` value that specifies the reason for signing the PDF document.

**subFilter**

Deprecated, use subFilterEx.

**subFilterEx**

A `PDFSubFilterType` value that represents an encoding to use when signing the PDF form. These string values are valid:

**adbe_pkcs7_detached:**

No data is encapsulated in the PKCS#7-signed data field.

**adbe_pkcs7_sha1:**

The adbe.pkcs7.sha1 digest of the byte range is encapsulated in the PKCS#7-signed data field.

**adbe_x509_rsa_sha1:**

The adbe.x509.rsa.sha1 digest uses the RSA encryption algorithm and SHA-1 digest method.

**timeStampSeed**

A `PDFTimeStampSeed` value that represents the timestamp associated with this seed value dictionary. It contains two values:

**url:**

A `string` value that represents the URL of the timestamp server used. If the URL is `null` and a timestamp is required, the URL is obtained from the certificate that is used to sign the PDF document.

**requiresTimeStamp:**

A `boolean` value that specifies whether the timestamp is required while signing a PDF document.

**version**

A `double` value that specifies the minimum PDF version required to sign the signature field. The valid values are PDF 1.5 and PDF 1.7.

## Datatype specific settings

Properties for the document signature.

For the properties that are formatted as an editable list, use the following buttons to manage the list:

**Add A List Entry:**

Adds an entry to the list. Depending on the option, type the information, select an item from a drop-down list, or select a file from a network location or computer. When you select a file from a location on your computer, during run time, the file must exist in the same location on AEM Forms Server.

**Delete Selected List Entry:**

Removes an entry from the list.

**Move Selected List Entry Up One Row:**

Moves the selected entry up in the list.

**Move Selected List Entry Down One Row:**

Moves the selected entry down in the list.

Some properties have the Required option beside them. Selecting this option means that the property is a required constraint and without it, the signing fails.

**Signature Handler Options**

Options for specifying the filters and subfilters used for validating a signature field. The signature field is embedded in a PDF document and the seed value dictionary is associated with a signature field.

*Signature Handler*

A list of handlers to use for the digital signatures. Adobe.PPKLite is a string valid value that can be selected to represent the creation and validation of Adobe-specific signatures. You can use other signature handlers by typing values, such as `Entrust.PPEF`, `CIC.SignIt`, and `VeriSign.PPKVS`. For information about supported signature handlers, see PDF Utilities. No default value is selected.

**Adobe.PPKLite:**

The recommended handler for signing PDF documents.

**Required:**

Select to specify that the signature handler is used for the seed value. It is not selected by default.

*Signature SubFilter*

The supported subfilter names, which describes the encoding of the signature value and key information. Signature handlers must support the listed subfilters; otherwise, the signing fails. These string values are valid for public-key cryptographic (See PDF Utilities.), which you must type:

**adbe.x509.rsa_sha1:**

The key contains a DER-encoded PKCS#1 binary data object. The binary objects represent the signature obtained as the RSA encryption of the byte range SHA-1 digest with the private key of the signer. Use this value when signing PDF documents using PKCS#1 signatures.

**adbe.pkcs7.detached:**

The key is a DER-encoded PKCS#7 binary data object containing the signature. No data is encapsulated in the PKCS#7-signed data field.

**adbe.pkcs7.sha1:**

The key is a DER-encoded PKCS#7 binary object representing the signature value. The SHA-1 digest of the byte range digest is encapsulated in the PKCS#7 signed data.

**Required:**

Select to specify that signature subfilters are used for the seed value. It is not selected by default.

*Digest Methods*

The list of acceptable hashing algorithms to use. No default hashing algorithm is provided. Add an item to the list and select an encryption algorithm. Select one of these values:

**SHA1:**

The Secure Hash Algorithm that has a 160-bit hash value.

**SHA256:**

The Secure Hash Algorithm that has a 256-bit hash value.

**SHA384:**

The Secure Hash Algorithm that has a 384 bit-hash value.

**SHA512:**

The Secure Hash Algorithm that has a 512 bit-hash value.

**RIPEMD160:**

The RACE Integrity Primitives Evaluation Message Digest that has a 160-bit message digest algorithm and is not FIPS-compliant.

**Required:**

Select to specify that the signature encryption algorithms are used for the seed value. It is not selected by default.

*Minimum Signature Compatibility Level*

The minimum PDF version to use to sign the signature field. No default value is selected. Select one of these values:

**PDF 1.5:**

Use PDF Version 1.5.

**PDF 1.7:**

Use PDF Version 1.7.

**Required:**

Select to specify the minimum signature compatibility level is used for the seed value. It is not selected by default.

**Signature Information**

A group of options for specifying the reasons, timestamp, and details of the digital signature.

*Include Revocation Information In Signature*

Select to specify that revocation information must be embedded as part of the signature for long-term validation support. When you deselect this option, the revocation information is not embedded as part of the signature. By default, this option is deselected.

**Required:**

Select to specify that revocation checking is required for the seed value. It is not selected by default.

*Signing Reasons*

The list of reasons that are associated with the seed value dictionary used for signing the PDF document. Add an item to the list and type a reason.

**Required:**

Select to specify that the associated reasons are included for the seed value. It is not selected by default.

*TimeStamp Server URL*

The URL that specifies the location of the timestamp server to use when signing a PDF document.

**Required:**

Select to specify that the timestamp server is required for the seed value. It is not selected by default.

*Signing/Enrollment Server URL*

The location of the server that provides a web service. The web service digitally signs a PDF document or enrolls for new credentials.

**Required:**

Select to specify that the signing or enrollment server is used for the seed value. It is not selected by default.

*Server Type*

The type of server to use for the value specified for the Signing/Enrollment Server URL option. The default value is Browser. Select one of these values:

**Browser:**

The URL references content that is displayed in a web browser to allow enrolling for a new credential if a matching credential is not found.

**ASSP:**

The URL references a signature web service. The web service is used to digitally sign the PDF document on a server. The server is specified in the Signing/Enrollment Server URL option in this operation.

**Required:**

Select to use the web service to sign the PDF document. It is not selected by default.

**Signature Type**

The changes that are permitted after the signature is added and legal attestations are provided.

*Type of Signature*

The list representing the type of signatures that can be applied to the signature field. The default value is Any. Select one of these values.

**Any:**

Any type of signature can be applied when filling in forms, instantiating page templates, or creating, deleting, and modifying annotations.

**Recipient Signature:**

Constrains the signer to apply a Four Corner security model on the signature field.

**Certification Signature:**

Constrains the signer to apply a certification signature on the signature field with specified permissions. The specified permissions are configured in the Field MDP Options Spec property for this operation. No default value is selected. Select one of these values:

- **No changes allowed**: The end user is not permitted to change the form. Any change invalidates the signature.
- **Form fill-in and digital signatures**: The end user is permitted to fill in the form, instantiate page templates, and sign the form.
- **Annotations, form fill-in, and digital signatures**: The end user is permitted to fill in the form, instantiate page templates, sign the form, and create annotations, deletions, and modifications.

*Legal Attestations*

The list of legal attestations associated with the seed value. Legal attestation constraints affect only a certification signature. Add a legal attestation to the list by typing it. No default legal attestations are provided.

**Required:**

Select to specify that legal attestations are used for the seed value. It is not selected by default.

**Signing Certificates**

The list of certificates, keys, issuers, and policies used for a digital signature. Add certificates, keys, issuers, and policies to the list using the Open dialog box.

*Signing Certificates*

A list of certificates used for certifying and verifying a signature.

**Required:**

Select to specify that signing certificates are used for the seed value. It is not selected by default.

*Subject Distinguished Name*

The list of dictionaries, where each dictionary contains key value pairs that specify the subject distinguished name (DN). The DN must be present within the certificate for it to be acceptable for signing. Add DNs to the list by using the Add Subject DN dialog box. (See AboutAdd Subject DN.)

**Required:**

Select to specify that subject distinguished names are used for the seed value. It is not selected by default.

*KeyUsage*

The list of key usage extensions that must be present for signing a certificate. Add an entry to the list and select the key usage. The default for both the DigitalSignature field and Non-Repudiation field is Don't Care:

**Don't Care:**

The key usage extension is optional.

**Require Key Usage:**

The key usage extension must be present.

**Exclude Key Usage:**

The key usage extension must not be present.

**Required:**

Select to specify that key usage extensions are used for the seed value. It is not selected by default.

Additional key usage entries are available in the PDF Utilities.

**Issuers and Policies**

The list of certificate issuers, policies, and associated object identifiers.

*Certificate Issuers*

The list of certificate issuers. Add certificate issuers to the list using the Open dialog box.

**Required:**

Select to specify that certificate issuers are used for the seed value. It is not selected by default.

*Certificate Policies and Associated Object Identifiers*

The list certificate policies associated with the certificate seed value. Add certificate policies to the list by typing it.

**Required:**

Select to specify that certificate policies and associated identifies are used for the seed value. It is not selected by default.

# 21.93. PDFSignature

A complex data type that is used to store the results of the GetSignature operation that the Signature service provides. It contains the signature and information about the filter and subfilters that are used for validating the signature embedded in the PDF document.

For information about data that can be accessed using Xpath Expressions, see Dataitems.

## Data items

The data items that `PDFSignature` variables contain.

### certificates

A `list` of `byte` values that represents the certificates that are embedded in the PDF document at the time of signing.

### contents

A `list` of `byte` values that represents the actual signature value in the form of a DER-encoded PKCS#1 or PKCS#7. If `subfilter` is set as adbe.pkcs7.detached or adbe.pkcs7.sha1, the PKCS#7 packet contains the embedded certificates.

### CRLs

A `list` of `byte` values that represents Certificate Revocation List (CRL) information.

### filter

A `string` value that represents the name of the preferred signature handler to use for validating the signature. For example, preferred signature handlers can be Adobe.PPKLite, Entrust.PPKEF, CICI.SignIt, and VeriSign.PPKVS.

**OCSPResponses**

A `list` of `byte` values that contains information returned from the Online Certificate Status Protocol (OCSP) server.

**subFilter**

A `string` value that represents the encoding to use when signing the PDF form. These values are valid:

**adbe_pkcs7_detached:**

No data is encapsulated in the PKCS#7-signed data field.

**adbe_pkcs7_sha1:**

The adbe.pkcs7.sha1 digest of the byte range is encapsulated in the PKCS#7-signed data field.

**adbe_x509_rsa_sha1:**

The adbe.x509.rsa.sha1 digest uses the RSA encryption algorithm and SHA-1 digest method.

**timestamp**

A `list` of `byte` values that contains information returned from the time-stamping provider (TSP).

# 21.94. PDFSignatureAppearanceOptionSpec

A complex data type that represents appearance aspects of a PDF signature field, used with the CertifyPDF operation of the Signature service. For example, you can specify that the name of the signer and the date must be displayed within the signature.

For information about data that can be accessed using Xpath Expressions, see Dataitems.

For information about configuring default properties, see Datatypespecific settings.

## Data items

The data items that `PDFSignatureAppearanceOptionSpec` variables contain.

**appearanceType**

A `string` value with a finite number of valid values that specify the appearance type of the signature. This value is used only for the visible signature fields. This value is ignored for invisible signature fields. The default value is `Name`.

These string values are valid:

**No Graphic:**

The appearance of the signature consists of only the signature text.

**Graphic:**

The appearance of the signature consists of a graphic area and a text area. The graphic area displays the PDF document specified by the Graphic PDF Document option. The text area displays the signature text.

**Name:**

The appearance of the signature consists of a graphic area and a text area. The graphic area displays a graphic of the name of the signer and the text area displays the signature text.

### graphicPDF

A `document` value that represents the graphic that appears within the signature when the Graphic signature type is configured. The value can be PDF documents only.

### logoOpacity

A `double` value that represents the opacity of the logo that appears within the signature. The valid values are between `0.0` (fully transparent) and `1.0` (fully opaque). The default value is `0.5`.

### logoPDF

A `document` value that represents the PDF document that appears within the signature. The value can be PDF documents only.

### showDate

A `boolean` value that indicates whether the date is displayed within the signature. The default value of `true` indicates that the date is displayed. A value of `false` indicates that the date is not displayed.

### showDefaultLogo

A `boolean` value that specifies whether the default Adobe logo is displayed within the signature. The default value of `true` indicates that the default Adobe logo is displayed within the signature. A value of `false` indicates that the Adobe logo is not displayed.

### showDN

A `boolean` value that indicates whether the Distinguished Name (DN) of the signer certificate is shown in the signature. The default value of `true` indicates that the DN is displayed. A value of `false` indicates that the DN is not displayed.

### showLabels

A `boolean` value that indicates whether labels for the various displayed items are displayed within the signature. The default value of `true` indicates that the labels are displayed. A value of `false` indicates that the labels are not displayed.

### showLocation

A `boolean` value that specifies whether the specified location is displayed within the signature. The default value of `true` indicates that the location is displayed. A value of `false` indicates that the location is not displayed.

### showName

A `boolean` value that specifies whether the name of the signer is displayed within the signature. The default value of `true` indicates that the name is displayed. A value of `false` indicates.

### showReason

A `boolean` value that specifies whether the reason for signing the PDF document is displayed within the signature. The default value of `true` indicates that the reason is displayed. A value of `false` indicates that the reason is not displayed.

### textDirection

A `TextDirection` value that specifies the direction of the text displayed within the signature. These values are valid:

**Auto:**

The direction of the text is based on the auto values.

**Left:**

The direction of the text is from left to right.

**Right:**

The direction of the text is from right to left.

## Datatype specific settings

Properties for configuring appearance aspects of a PDF signature.

### Signature Type

Sets the appearance type of the signature. The default value is Name. Select one of these values:

**No Graphic:**

The appearance of the signature consists of only the signature text.

**Graphic:**

The appearance of the signature consists of a graphic area and a text area. The graphic area displays the PDF document specified by the Graphic PDF Document option. The text area displays the signature text.

**Name:**

The appearance of the signature consists of a graphic area and a text area. The graphic area displays a graphic of the name of the signer and the text area displays the signature text.

### Graphic PDF Document

Sets the graphic that is displayed within the signature if a signature type of Graphic is used. Only a PDF file can be used. This option can be set if Graphic is selected in the Signature Type list.

When you click the ellipsis button ..., the Open dialog box opens. In the dialog box, you can select a file from your computer or from a network location. During run time, if you selected a file from a location on your computer, it must exist in the same location on AEM Forms Server.

### Use Default Adobe PDF Logo

Select this option to display the default Adobe PDF logo within the signature appearance. When this option is deselected, the Adobe logo is not displayed. By default, the option is selected.

### Logo PDF Document

Sets a PDF document to display within the signature appearance. The PDF document contains an image to display. This option can be set when the Use Default Adobe PDF Logo option is deselected.

When you click the ellipsis button, the Open dialog box opens. In the dialog box, you can select a file from your computer or from a network location. During run time, if you selected a file from a location on your computer, it must exist in the same location on AEM Forms Server.

### Logo Opacity

Sets the opacity of the logo that is displayed within the signature. Valid values are from `0.0` (fully transparent) to `1.0` (fully opaque). Can be set only if Use Default Adobe PDF Logo is deselected. If any value outside this range is specified, the default of `0.50` is used.

### Text Direction

Sets the direction of the text displayed within the signature. The default value is Auto. Select one of these values:

**Auto:**

Use the direction specified by the PDF document.

**Left:**

The text direction is left to right.

**Right:**

The text direction is right to left.

### Show Name

Select this option to display the name of the signer in the digital signature. When this option is deselected, the name of the signer is not displayed. By default, the option is selected.

### Show Reason

Select this option to display the reason the PDF document was signed in the digital signature. When this option is deselected, the reason is not displayed. By default, the option is selected.

### Show Distinguished Name

Select this option to display the certificate of the signer in the digital signature. When this option is deselected, the certificate is not displayed. By default, the option is selected.

### Show Date

Select this option to display the date the PDF document was signed in the digital signature. When this option is deselected, the date is not displayed. By default, the option is selected.

### Show Location

Select this option to display the location the PDF document was signed in the digital signature. When this option is deselected, the location is not displayed. By default, the option is selected.

### Show Labels

Select this option to display the labels for the preceding display items. When this option is deselected, the labels for the preceding display items are not displayed. By default, the option is selected.

# 21.95. PDFSignatureField

A complex data type that stores the results of the GetSignature Field List and Get CertifyingSignature Field operations that the Signature service provides. It contains information about the signature fields located within the PDF document.

For information about data that can be accessed using Xpath Expressions, see Dataitems.

## Data items

The data items that `PDFSignatureField` variables contain.

### name

A `string` value that represents the fully qualified name of the signature field. If you are signing a PDF document that is based on a form created in Designer, you can use a partial name of the signature field. For example, `form1[0].#subform[1].SignatureField3[3]` can be specified as `SignatureField3[3]`.

**properties**

A `PDFSignatureFieldProperties` value that represents the properties of the signature field. These values are valid:

**fieldMDP:**

Specifies the type of fields that are locked when the signature field is signed.

**SeedValue:**

Constraining information that is used at the time the signature is applied.

**signed**

A `boolean` value that indicates whether the signature field is signed or certified. A value of `true` indicates that the signature field is signed and `false` indicates that the signature field is certified.

**type**

A `PDFSignatureType` value that represents the type of signature field.

**visible**

A `boolean` value that indicates whether the signature field is visible. A value of `true` indicates that the field is visible and `false` indicates that the signature field is hidden.

# 21.96. PDFSignatureFieldProperties

A complex data type used by the ModifySignature Field operation provided by the Signature service. It contains information about the signature fields in a PDF document. The information includes which signature fields are locked when the signature field is signed, and constraining information that is used at the time the signature is applied.

For information about data that can be accessed using Xpath Expressions, see Dataitems.

For information about configuring default properties, see Datatypespecific settings.

## Data items

The data items that `PDFSignatureFieldProperties` variables contain.

**fieldMDP**

A `FieldMDPOptionSpec` value that specifies the fields that are locked after the signature field is signed. These values are valid:

**action:**

The locking action that occurs when the signature field is signed.

**fields:**

A list of field names to which action applies.

**seedValue**

A `PDFSeedValueOptionSpec` value that contains constraining information used at the time the signature is applied.

## Datatype specific settings

Properties for configuring the appearance aspects of a PDF Signature Field.

**Field MDP Options Spec**

Specifies the PDF document fields that are locked after the signature field is signed.

*Field Locking Action*

A list that sets the type of action to use to lock fields in a PDF document. No default value is selected. Select one of these values:

**All Fields:**

Lock all fields in the PDF document.

**Include Fields:**

Lock only the fields specified in the Application To Form Fields option.

**Exclude Fields:**

Lock all fields except for those fields specified in the Applicable To Form Fields option.

*Applicable to Form Fields*

Sets a comma-separated list of field names that indicate which fields the action is applicable or not applicable to. This option is available when Field Locking Action option is set to a value of Include Fields or Exclude Fields.

**Seed Value Options Spec**

Properties for the document signature.

For the properties that are formatted as an editable list, use the following buttons to manage the list:

**Add A List Entry:**

Adds an entry to the list. Depending on the option, type the information, select an item from a drop-down list, or select a file from a network location or computer. When you select a file from a location on your computer, during run time, the file must exist in the same location on AEM Forms Server.

**Delete Selected List Entry:**

Removes an entry from the list.

**Move Selected List Entry Up One Row:**

Moves the selected entry up in the list.

**Move Selected List Entry Down One Row:**

Moves the selected entry down in the list.

Some properties have the Required option beside them. Selecting this option means that the property is a required constraint and without it, the signing fails.

**Signature Handler Options**

Options for specifying the filters and subfilters used for validating a signature field. The signature field is embedded in a PDF document and the seed value dictionary is associated with a signature field.

*Signature Handler*

A list of handlers to use for the digital signatures. Adobe.PPKLite is a string valid value that can be selected to represent the creation and validation of Adobe-specific signatures. You can use other signature handlers by typing values, such as `Entrust.PPEF`, `CIC.SignIt`, and `VeriSign.PPKVS`. For information about supported signature handlers, see PDF Utilities. No default value is selected.

**Adobe.PPKLite:**

The recommended handler for signing PDF documents.

**Required:**

Select to specify that the signature handler is used for the seed value. It is not selected by default.

*Signature SubFilter*

The supported subfilter names, which describe the encoding of the signature value and key information. Signature handlers must support the listed subfilters; otherwise, the signing fails. These string values are valid for public-key cryptographic (see PDF Utilities), which you must type:

**adbe.x509.rsa_sha1:**

The key contains a DER-encoded PKCS#1 binary data object. The binary objects represent the signature obtained as the RSA encryption of the byte range SHA-1 digest with the private key of the signer. Use this value when signing PDF documents using PKCS#1 signatures.

**adbe.pkcs7.detached:**

The key is a DER-encoded PKCS#7 binary data object containing the signature. No data is encapsulated in the PKCS#7-signed data field.

**adbe.pkcs7.sha1:**

The key is a DER-encoded PKCS#7 binary object representing the signature value. The SHA-1 digest of the byte range digest is encapsulated in the PKCS#7 signed data.

**Required:**

Select to specify that signature subfilters are used for the seed value. It is not selected by default.

*Digest Methods*

The list of acceptable hashing algorithms to use. No default hashing algorithm is provided. Add an item to the list and select an encryption algorithm. Select one of these values:

**SHA1:**

The Secure Hash Algorithm that has a 160-bit hash value.

**SHA256:**

The Secure Hash Algorithm that has a 256-bit hash value.

**SHA384:**

The Secure Hash Algorithm that has a 384 bit-hash value.

**SHA512:**

The Secure Hash Algorithm that has a 512 bit-hash value.

**RIPEMD160:**

The RACE Integrity Primitives Evaluation Message Digest that has a 160-bit message digest algorithm and is not FIPS-compliant.

**Required:**

Select to specify that the signature encryption algorithms are used for the seed value. It is not selected by default.

*Minimum Signature Compatibility Level*

The minimum PDF version to use to sign the signature field. No default value is selected. Select one of these values:

**PDF 1.5:**

Use PDF Version 1.5.

**PDF 1.7:**

Use PDF Version 1.7.

**Required:**

Select to specify the minimum signature compatibility level is used for the seed value. It is not selected by default.

**Signature Information**

A group of options for specifying the reasons, timestamp, and details of the digital signature.

*Include Revocation Information In Signature*

Select to specify that revocation information must be embedded as part of the signature for long-term validation support. When you deselect this option, the revocation information is not embedded as part of the signature. By default, this option is deselected.

**Required:**

Select to specify that revocation checking is required for the seed value. It is not selected by default.

*Signing Reasons*

The list of reasons that are associated with the seed value dictionary used for signing the PDF document. Add an item to the list and type a reason.

**Required:**

Select to specify that the associated reasons are included for the seed value. It is not selected by default.

*TimeStamp Server URL*

The URL that specifies the location of the timestamp server to use when signing a PDF document.

**Required:**

Select to specify that the timestamp server is required for the seed value. It is not selected by default.

*Signing/Enrollment Server URL*

The location of the server that provides a web service. The web service digitally signs a PDF document or enrolls for new credentials.

**Required:**

Select to specify that the signing or enrollment server is used for the seed value. It is not selected by default.

*Server Type*

The type of server to use for the value specified for the Signing/Enrollment Server URL option. The default value is Browser. Select one of these values:

**Browser:**

The URL references content that is displayed in a web browser to allow enrolling for a new credential if a matching credential is not found.

**ASSP:**

The URL references a signature web service. The web service is used to digitally sign the PDF document on a server. The server is specified in the Signing/Enrollment Server URL option in this operation.

**Required:**

Select to use the web service to sign the PDF document. It is not selected by default.

**Signature Type**

The changes that are permitted after the signature is added and legal attestations are provided.

*Type of Signature*

The list representing the type of signatures that can be applied to the signature field. The default value is Any. Select one of these values.

**Any:**

Any type of signature can be applied when filling in forms, instantiating page templates, or creating, deleting, and modifying annotations.

**Recipient Signature:**

Constrains the signer to apply a Four Corner security model on the signature field.

**Certification Signature:**

Constrains the signer to apply a certification signature on the signature field with specified permissions. The specified permissions are configured in the Field MDP Options Spec property for this operation. No default value is selected. Select one of these values:

**No changes allowed:**

The end user is not permitted to change the form. Any change invalidates the signature.

**Form fill-in and digital signatures:**

The end user is permitted to fill in the form, instantiate page templates, and sign the form.

**Annotations, form fill-in, and digital signatures:**

The end user is permitted to fill in the form, instantiate page templates, sign the form, and create annotations, deletions, and modifications.

*Legal Attestations*

The list of legal attestations associated with the seed value. Legal attestation constraints affect only a certification signature. Add a legal attestation to the list by typing it. No default legal attestations are provided.

**Required:**

Select to specify that legal attestations are used for the seed value. It is not selected by default.

**Signing Certificates**

The list of certificates, keys, issuers, and policies used for a digital signature. Add certificates, keys, issuers, and policies to the list using the Open dialog box.

*Signing Certificates*

A list of certificates used for certifying and verifying a signature.

**Required:**

Select to specify that signing certificates are used for the seed value. It is not selected by default.

*Subject Distinguished Name*

The list of dictionaries, where each dictionary contains key value pairs that specify the subject distinguished name (DN). The DN must be present within the certificate for it to be acceptable for signing. Add DNs to the list by using the Add Subject DN dialog box. (See AboutAdd Subject DN.)

**Required:**

Select to specify that subject distinguished names are used for the seed value. It is not selected by default.

*KeyUsage*

The list of key usage extensions that must be present for signing a certificate. Add an entry to the list and select the key usage. The default for both the DigitalSignature field and Non-Repudiation field is Don’t Care:

**Don’t Care:**

The key usage extension is optional.

**Require Key Usage:**

The key usage extension must be present.

**Exclude Key Usage:**

The key usage extension must not be present.

**Required:**

Select to specify that key usage extensions are used for the seed value. It is not selected by default.

Additional key usage entries are available in the PDF Utilities.

**Issuers and Policies**

The list of certificate issuers, policies, and associated object identifiers.

*Certificate Issuers*

The list of certificate issuers. Add certificate issuers to the list using the Open dialog box.

**Required:**

Select to specify that certificate issuers are used for the seed value. It is not selected by default.

*Certificate Policies and Associated Object Identifiers*

The list certificate policies associated with the certificate seed value. Add certificate policies to the list by typing them.

**Required:**

Select to specify that certificate policies and associated identifies are used for the seed value. It is not selected by default.

# 21.97. PDFSignaturePermissions

A complex data type that represents the actions that can be performed in the PDF document without invalidating the signature. The `PDFSignaturePermissions` value is a member of the `PDFSignatureVerificationInfo` and `PDFSignatureVerificationResult` data types.

These string values are valid:

**All:**

All modifications to the PDF document are allowed.

**FormFields:**

Only modifications to the form fields are allowed.

**FormFieldsAndComments:**

Only modifications to the form fields and comments are allowed.

**None:**

Modifications to the PDF document are not allowed.

# 21.98. PDFSignatureType

A `string` value that represents the type of signature used to sign a PDF document. The `PDFSignatureType` data type is a member of `PDFSignatureField` and `SignatureType` data types.

These string values are valid:

**AUTHORSIG:**

The signature field is an author signature field.

**RECIPIENTSIG:**

The signature field is another type of signature field.

**XMLSIG:**

The signature field is an XML signature field.

**NOSIG:**

No signature field exists on the PDF document.

# 21.99. PDFSignatureVerificationInfo

A complex data type that is used to store the results of the VerifyPDF Signature operation that the Signature service provides. It contains verification information about the signature that is used to sign a PDF document.

For information about data that can be accessed using Xpath Expressions, see Dataitems.

## Data items

### fieldName

A `string` value that represents the name of the signature field on the PDF document.

### signatureProps

A `SignatureProperties` value that contains information about the PDF signature.

### signatureStatus

A `SignatureStatus` value that describes the signature status on the PDF document and dynamic XML forms. Dynamic XML forms are PDF forms created in Designer. The following string values are valid:

**SignatureFormatError:**

The signature is invalid because there are errors in the formatting or information contained in the signature.

**CertificationSigWithChanges:**

The changes that are made to the PDF document. The changes are permitted by the certifying entity that certified the PDF document.

**DocSigWithChanges:**

The revision of the PDF document or dynamic XML form that the signature covers has not been altered; however, there are subsequent changes to it.

**CertifiedDocSigWithChanges:**

Changes have been made to the PDF document or dynamic XML form since the signature was applied. The changes are permitted by the certifying entity that certified the PDF document.

**DocumentCertificationSigNoChanges:**

The PDF form has not been modified since the certification signature was applied.

**DocumentSignatureUnknown:**

The status of the PDF document is unknown because the signatures could not be verified.

**CertifiedDocumentSignatureTamper:**

The certification signature in the PDF document is altered or corrupt.

**DocumentSigNoChanges:**

The PDF form has not been modified since the signature was applied.

**SignedDocumentSignatureTamper:**

The signatures in the PDF document are altered or corrupt.

**DynamicFormSignatureUnknown:**

The status of the dynamic XML form is unknown because the signatures could not be verified.

**CertifiedDynamicFormSignatureTamper:**

The certification signature in the dynamic XML form is altered or corrupt.

**DynamicFormSigNoChanges:**

The dynamic XML form has not been modified since the signature was applied.

**SignedDynamicFormSignatureTamper:**

The signatures in the dynamic XML form are altered or corrupt.

**signatureType**

A `SignatureType` value that specifies whether a user or a certification signature signs the PDF document.

**permissions**

A `PDFSignatureVerificationResult` value that represents the actions that can be performed in the PDF document without invalidating the signature.

**type**

An `PDFSignatureType` value that specifies the type of signature that was applied to the PDF document.

**signer**

A `IdentityInformation` value that contains information about the identity of the signer of the PDF document.

# 21.100. PDFSignatureVerificationResult

A complex data type that is used to store the results of the VerifyPDF Signature (deprecated) operation that the Signature service provides. It contains information about the signature that is used to sign a PDF document, including its validity status.

For information about data that can be accessed using Xpath Expressions, see Dataitems.

## Data items

The data items that `PDFSignatureVerificationResult` variables contain.

**certPaths**

A `signerCertificatePath` value that represents a certificate path, information about the path's validation status, and the possible failure reason.

The `signerCertificatePath` variable contains these values:

**CertificateDER**

A `list` of `byte` value of certificates. Each certificate is represented by its DER encoding.

**FailureReason**

A `PathValidationFailureReason` value specifies the reason a path is not valid. These string values are valid:

- **CERTIFICATE_EXPIRED:** The path is not valid because the certificate expired.
- **IDENTRUS_OCSP_COMPLIANCE_FAILED**: The path is not valid because verification of IdenTrust OCSP compliance failed.
- **IDENTRUS_SIGNING_COMPLIANCE_FAILED**: The path is not valid because verification of Identrus compliance failed.
- **INCORRECT_SIGNATURE**: The path is not valid because the certificate signature is incorrect.

- **INVALID_BASIC_CONSTRAINTS**: The path is not valid because an invalid basic constraint was used.
- **INVALID_KEY_USAGE**: The path is not valid because an invalid key was used.
- **INVALID_NAME_CONSTRAINTS**: The path is not valid because an invalid basic constraint was used.
- **INVALID_POLICY_CONSTRAINTS**: The path is not valid because invalid policy constraints exist.
- **INVALID_POLICY_MAPPINGS**: The path is not valid because invalid policy mappings exist.
- **MISSING_BASIC_CONSTRAINTS**: The path is not valid because basic constraints are missing.
- **NO_FAILURE**: The path is valid.
- **PATH_LEN_CONSTRAINT_NOT_SATISFIED**: The path is not valid because the length constraint is not valid.
- **PATH_NOT_TRUSTED**: The path is not valid because the path is not trusted.
- **UNKNOWN_REASON**: The path is not valid for an unknown reason.
- **UNSUPPORTED_CRITICAL_EXTENSION**: The path is not valid because unsupported critical extensions exist.

### contactInfo

A `string` value that contains contact information of the signer.

### dateSigned

A `date` value that represents the date on which the PDF document was signed.

### legalAttestations

A `string` value that represents the legal attestations that are associated with the signature in the PDF document.

When a document is certified, it is automatically scanned for specific types of content. There are types of content that could potentially make the visible contents of a document ambiguous or misleading. For example, an annotation could obscure some text on a page that is important for understanding what is being certified. The scanning process generates warnings that indicate the presence of these types of content. This value provides an additional explanation of the content that generates warnings.

### location

A `string` value that represents the location of the signer.

### numRevisions

A `long` value that represents the number of revisions of the signature.

### permissions

A `DFSignatureVerificationResult`value that represents the actions that can be performed in the PDF document without invalidating the signature.

**policyQualifierList**

A `list` value that contains the policy qualifiers.

**reason**

A `string` value that represents why the PDF document was signed.

**revision**

An `int` value that represents the revision of the signature.

**signatureStatus**

A `PDFSignatureStatus` value that represents the status of the signature. The following are valid string values:

**Invalid:**

The signature is invalid. The revision of the document covered by the signature has been altered.

**Unknown:**

The status of the signature is unknown. The signature validation on the signed contents was not performed.

**ValidAndModified:**

The signature is valid but the document has been modified. The revision of the document covered by the signature was not modified, but there were subsequent changes to the document.

**ValidUnmodified:**

The signature is valid and the document is unmodified. The revision of the document covered by the signature was not modified. There were no subsequent changes to the document.

**signerName**

A `string` value that contains the signer name.

**signerStatus**

An `IdentityStatus` value that represents whether the signer is trusted. These string values are valid:

**NOTTRUSTED:**

This signer is not trusted or is invalid because the certificate is invalid or the certificate could not be chained back to a trusted root.

**TRUSTED:**

The signer is trusted because the certificate is both valid and can be chained back to a trusted root.

**UNKNOWN:**

This signer is unknown because verification of the signer could not be performed.

**signingDateTimestamped**

A `boolean` value that indicates whether the signature is date-stamped. A value of `true` indicates that the signature is date-stamped and `false` indicates that the signature is not date-stamped.

**timestamp**

A `byte array` value that contains the timestamp information.

**TSAInfo**

A `string` value that contains the timestamp information associated with the signature.

**TSAStatus**

An `IdentityStatus` value that represents whether the timestamp authority validation status is trusted. A timestamp specifies that specific data was established before a certain trusted time. A timestamp also contributes toward the process of building a trust relationship between the signer and verifier. These string values are valid:

**NOTTRUSTED:**

This signer is not trusted or is invalid because the certificate is invalid or the certificate could not be chained back to a trusted root.

**TRUSTED:**

The signer is trusted because the certificate is both valid and can be chained back to a trusted root.

**UNKNOWN:**

This signer is unknown because verification of the signer could not be performed.

# 21.101. PDFUtilitySaveMode

A complex data type that represents the save mode to be used by the PDF Utilities service when saving a PDF document.

For information about data that can be accessed using Xpath Expressions, see Dataitems.

## Data items

The data items that `PDFUtilitySaveMode` variables contain.

### required

A `boolean` value that indicates whether the save mode of the PDF document is required. A value of `true` indicates that the save mode is required, and `false` indicates that the save mode is not required.

### saveStyle

A `string` value that represents the save mode of a PDF document. These values are valid:

**FAST_WEB_VIEW:**

In addition to performing a FULL save, also restructures the PDF to accommodate page-at-a-time downloading from a web server. This save type invalidates signature certification and reader-enabled rights.

**INCREMENTAL:**

The saving operation is performed in the least amount of time. This save is performed only when the version of the baseDocument is 1.4 or later. Otherwise. a FULL save is performed.

**FULL:**

The saving operation is performed with fewer optimizations. The existing incremental saves are removed and the web viewing optimization is not preserved. This save type invalidates signature certification and reader-enabled rights.

# 21.102. PolicySpec

A complex data type used to specify a policy identifier. `PolicySpec` values are used to configure Getpolicy by policy identifier operations that the Rights Management service provides.

For information about data that can be accessed using Xpath Expressions, see Dataitems.

## Data items

The data items that `PolicySpec`variables contain.

### accessDeniedErrorMessage

A `string` value that represents the access denied error message for the policy.

### alternateId

A `string` value that represents the alternate identification for the policy.

### autoGenerated

A `boolean` value that specifies whether the policy is auto-generated.

**creationTime**

A `date` value that represents when the policy was first registered with the Rights Management service.

**deleted**

A `boolean` value that specifies whether the policy is deleted.

**description**

A `string` value that represents the description of the policy.

**encryptionAlgorithmAndKeySize**

A `string` value that represents the encryption algorithm and key size for the policy.

**lastUpdateTime**

A `date` value that represents when the policy was last updated.

**name**

A `string` value that represents the name of the policy.

**offlineLeasePeriod**

An `int` value that represents the offline lease period for the policy.

**owner**

An `User` value that represents the owner of the policy.

**policyId**

A `string` value that represents the identifier of the policy.

**policySetName**

A `string` value that represents the name of the set to which the policy belongs.

**policyType**

A `string` value that represents the policy type.

**policyXml**

A `byte` value that is an XML representation of the policy. The structure of the XML is defined by the PDRL schema.

**principals**

A `list` of com.adobe.idp.um.api.infomodel.Principal objects associated with the policy.

### watermarkName

A `string` value that represents the name of the watermark the policy uses.

# 21.103. PositionRectangle

A complex data type that contains information for configuring the Position Rectangle property of the AddVisible Signature Field operation that the Signature service provides. The data type represents the position and size of the signature field rectangle in a PDF document. The specified rectangle must be located, at least partially, within the page's crop box. The lower-left x and y values are relative to the crop box of the page.

For information about data that can be accessed using Xpath Expressions, see Dataitems.

For information about configuring default properties, see Datatypespecific settings.

## Data items

The data items that `PositionRectangle` variables contain.

### height

An `int` value that represents the height of the signature field. This value must be greater than zero.

### lowerLeftX

An `int` value that represents the lower-left x-coordinate of the signature field. It cannot be a negative value.

### lowerLeftY

An `int` value that represents the lower-left y-coordinate of the signature field. It cannot be a negative value.

### width

An `int` value that represents the width of the signature field. This value must be greater than zero.

## Datatype specific settings

Properties for configuring the position of the signature field rectangle.

### Lower Left X

Sets the lower-left X position of the signature field. The value cannot be zero or a negative number and is relative to the crop box of the page. The default value is `1`.

**Lower Left Y**

Sets the lower-left X position of the signature field. The value cannot be zero or a negative number and is relative to the crop box of the page. The default value is `1`.

**Height**

Sets the height of the signature field. The value cannot be zero or a negative number. The default value is `100`.

**Width**

Sets the width of the signature field. The value cannot be zero or a negative number. The default value is `100`.

# 21.104. Principal Reference

A complex data type that represents a principal. A principal can be either a user or group. `Principal Reference` values are used to configure Createpolicy from template (deprecated) operations that the Rights Management service provides.

For information about data that can be accessed using Xpath Expressions, see Dataitems.

## Data items

The data items that `PrincipalReference` variables contain.

**canonicalname**

A `string` value that represents the canonical name for the principal.

**commonName**

A `string` value that represents common name for the principal.

**domainCommonName**

A `string` value that represents the display name of the domain.

**domainName**

A `string` value that represents the canonical name of the domain.

**email**

A `string` value that represents the primary email identifier associated with the principal.

**oid**

A `string` value that represents the object identifier of the principal.

**org**

A `string` value that represents the organization that the principal belongs to.

**principalType**

A `string` value that represents the principal type. These values are valid:

**PRINCIPALTYPE_GROUP:**

A group.

**PRINCIPALTYPE_USER:**

A user.

**PRINCIPALTYPE_SYSTEM:**

A system principal that is used for hidden accounts and do not run as services.

**PRINICIPALTYPE_SERVICE:**

A system principal that is used to run services continually.

**status**

A `string` value that represents the status of a principal. These values are valid:

**STATUS_CURRENT:**

The principal is current and active.

**STATUS_OBSOLETE:**

The principal is obsolete and deleted.

**system**

A `boolean` value that indicates whether the user or group is a system principal. A value of `True` specifies that the principle is a system principle. A value of `False` specifies that the principal is not a system principal.

**visibility**

An `int` value that represents the visibility level of the principal. These values are valid:

**0:**

The principal is invisible.

**1:**

The principal can be found with searches.

**2:**

The principal can appear in the user interface.

# 21.105. PrintedOutputOptionsSpec

A complex data type that represents Output service options for tasks such as sending output to a network printer, an email recipient as a file attachment, or a file such as a PostScript file. `PrintedOutputOptionsSpec` variables are used to configure the Printed Output Options property in the generatePrintedOutput (deprecated) operation of the Output service.

For information about data that can be accessed using Xpath Expressions, see Dataitems.

For information about configuring default properties, see Datatypespecific settings.

## Data items

The data items that `PrintedOutputOptionsSpec` variables contain.

### charset

A `string` value that represents the character set to use. For a list of character sets, see http://java.sun.com/j2se/1.5.0/docs/guide/intl/encoding.doc.html.

### copies

An `int` value that represents the number of copies that the Output service creates during printing. A positive integer must be specified. The default value is 1. If an invalid value is provided, the default value of 1 replaces the invalid value.

### debug

A `boolean` value that determines whether debug-level logging is performed. A value of `true` indicates that debug-level logging is performed, and `false` indicates that the default level of logging is performed.

### fileURI

A `string` value that represents the URI of a file to send the output.

### generateManyFiles

A `boolean` value that specifies whether the Output service creates single or multiple output. A value of `true` indicates that multiple output are generated and `false` indicates a single output is generated.

### lazyloading

A `boolean` value that specifies whether incremental (lazy) loading is used when processing multi-record data sets. When set to the a value of `true`, multi-record data sets are loaded and merged one record of data at a time. Processing one record at a time helps to avoid running out of memory. The use of incremental loading limits XLST options specified in the XCI file because transformations can only be applied to only one record.

### locale

A `string` value that represents the locale to use. For a list of supported locale codes, see http://java.sun.com/j2se/1.5.0/docs/guide/intl/locale.doc.html.

### lookAhead

An `int` value that represents the number of bytes that are used from the beginning of the input data file to scan for the pattern strings. The default value is `500`.

### lpdURI

A `string` value that represents the Line Printer Daemon (LPD) URI of an LPD that is running on the network.

### metaDataSpecFile

A `boolean` value that indicates whether metadata is generated. A value of `true` indicates that metadata is generated and `false` indicates that metadata is not generated.

### outputBin

A `string` value that represents the XCI value that is used to enable the print driver to select the appropriate output bin.

### outputJog

A `OutputJog` value that indicates which output pages are physically shifted in the output tray. This data item is for use with PS and PCL printers only.

### pageOffsetX

A `string` value that represents the page offset to use in the X-direction of the paper on which the output is printed. This value is useful when for printing on preprinted paper. This value overrides the page offsets defined in the XDC file, which represents non-printable area for any PCL device. This option is for PCL devices only.

### pageOffsetY

A `string` value that represents the page offset to use in the Y-direction of the paper that the output is printed on. This value is useful for printing on preprinted paper. This value overrides the page offsets

defined in the XDC file, which represents non-printable area for any PCL device. This option is for PCL devices only.

**pagination**

A `Pagination` value that represents the pagination to use for the output.

**printerQueueName**

A `string` value that represents the name of the printer queue that is used with the Line Printer Daemon (LPD) URI.

**printerURI**

A `string` value that specifies a destination for the print output. A value that represents the name of a network printer causes the Output service to send the print stream to a printer.

**recordIdField**

A `string` value that represents the name of the element that contains a batch record in the input data file.

**recordLevel**

An `int` value that represents the level of the XML element that contains a batch record. The batch record level is the element level (located within the input data file) that contains data records. The root element of the XML document is level 1. Do not specify a value for this data item if you specify a value for the recordName property.

**recordLevelMetaData**

A `boolean` value that specifies whether the Output service generates metadata. A value of `true` indicates metadata is generated and `false` indicates that metadata is not generated.

**recordName**

A `string` value that represents the element name that identifies the beginning of a batch of records. The element name is located within the input data file. Any value indicates that the input data contains record batches. Do not specify a value for this data item if you specify a value for the recordLevel item.

**rules**

A `list` of `string` values that represents search rules that scan the input data file for a pattern. This data item also associates the data with a specific form design.

**serverPrintSpec**

A `string` value that represents the filename of a print specification.

### staple

A `Staple` value that specifies whether a stapler is used for printer output. Use this option for only PS and PCL printers.

### XCIURI

A `string` value that specifies the XCI file to use. By default, the Output service uses an XCI file named pa.xci. The value in this data item overrides the default file.

## Datatype specific settings

Properties to configure the options to use for rendering a printed PDF document.

### General

*XCI URI*

Sets the XCI file to use. XCI files are used to describe fonts that are used for elements in a form design. XCI files are also useful for specifying print options such as number of copies, whether duplex printer is used, or stapler options.

*Character Set*

Sets the character set used to encode the rendered form. Select the character set to use or one of these values.

**<Use Server Default>:**

(Default) Use the Character Set setting that is configured on AEM Forms Server. The Character Set setting is configured using Administration Console. (See AEM Forms administration help.)

**<Use Custom Value>:**

Use a character set that is not in the list. After you select this value, in the box beside the list, type the canonical name of the encoding set that is not available. For a list of character sets, see http://java.sun.com/j2se/1.5.0/docs/guide/intl/encoding.doc.html.

*Locale*

Sets the language used for generating the PDF document. Select a language from the list or one of these values.

**<Use Server Default>:**

(Default) Use the Locale setting that is configured on AEM forms Server. The Locale setting is configured using administration console. (See AEM Forms administration help.)

**<Use Custom Value>:**

Use a locale that is not in the list. After selecting this value, in the box beside the list, type the Locale ID of the locale code to use. For a complete list of supported locale codes, see http://java.sun.com/j2se/1.5.0/docs/guide/intl/locale.doc.html.

**Batch**

*Record Name*

Sets the name of the element that identifies the beginning of a batch of records.

*Record Level*

Sets the XML element level that contains the record data. The default is 1, which represents the first level of the record that contains data. The first level is typically below the element specified by the Record Name property.

*Generate Multiple Streams*

Sets whether the operation creates a single or multiple output for each record that is merged with a form design. Select one of these values:

**True:**

Creates multiple outputs.

**False:**

(Default) Creates a single output.

*Enable Lazy Loading*

Sets whether incremental (lazy) loading is used when processing multi-record data sets. Select one of these values:

**True:**

Multi-record data sets are loaded and merged one record of data at a time. Processing one record at a time helps to avoid running out of memory, but limits the use of XSLT in the XCI file. The use of incremental loading limits XLST options specified in the XCI file because transformations can only be applied to only one record.

**False:**

All records are loaded and merged at one time because the entire data file is loaded.

**Rules**

*Pattern Match Size*

Sets the number of bytes to use from the beginning of the input data file to scan for the pattern strings. The default is 500.

*Pattern Matching Rules*

Sets rules for scanning the input data file for a pattern and associates the data with a specific form design.

**Add A List Entry:**

Adds a new rule. After you click this button, a new entry is created in the list. In the Pattern field for the new entry, type a pattern to search for. In the Form field, type the name of the form design for the matching pattern. All the form designs that you specify must be available at the location specified by the Content Root property in this operation.

**Delete A Selected List Entry:**

Removes the selected rule from the list.

For information about working with search rules, Designer Help.

**Destination**

*Output Location URI*

Sets the URI of the path and file to save the output file to. If you create multiple files, the filenames are suffixed with a numeric value.

*Printer Name*

Sets the name of the printer to send the output to for printing.

*LPD URI*

Sets URI of the Line Printer Daemon (LPD) to use.

*LPD Printer Name*

Sets the name of the printer on the specified Line Printer Daemon (LPD) URI to use.

**Printer**

*Duplex Printing*

Sets to use single-sided or two-sided printing. Select one of these values:

**No (Simplex):**

Use single-sided printing.

**Duplex Long Edge:**

Use two-sided printing and print using long-edge pagination.

**Duplex Short Edge:**

Use two-sided printing and print using short-edge pagination.

*Staple*

Sets whether to staple the output on printer. Select of these values:

**Off:**

Do not use the stapler on the printer.

**On:**

Use the stapler on the printer.

**Use Printer Setting:**

Use printer stapler settings.

*Page Offset X*

Sets the page offset in the horizontal direction. This value is useful when you print the job on preprinted paper and change the origin (for example, the default page offsets for that particular job). This value overrides the page offsets defined in the XDC file. The XDC file settings represent a non-printable area for any PCL device. This option is for PCL devices only.

Type a value and beside the box, select in (inches), mm (millimeters), cm (centimeters), or pt (points).

*Page Offset Y*

Sets the page offset in the vertical direction. This value is set when you print the job on preprinted paper and change the origin (for example, the default page offsets for that particular job). This value overrides the page offsets defined in the XDC file. The XDC file represents a non-printable area for any PCL device. This option is for PCL devices only.

*Output Bin*

Sets the output tray on the printer. For example, type `outputtray1` to specify the first output tray on a printer. This value is used to enable the print driver to select the appropriate output bin.

*Output Jog*

Sets that the output pages that are physically shifted in the output tray. This option is only for printers that support PS and PCL. Select one of these values:

**Use Printer Setting:**

Use the jog option setting configured on the printer.

**None:**

Do not use jogging.

**PageSet:**

Use jog each time a set of pages are printed.

### MetaData

*Meta Data Spec File*

Sets the URI of the metadata spec file to use. A metadata spec file is used to generate metadata from the provided data file.

*Record ID XPath*

Sets the root level node to use for XPath expressions. The root level node specifies the starting point for XPath expressions.

*Generate Record Level Meta Data*

Sets whether to generate a metadata file for each record. Select one of these values:

**True:**

Generate a metadata file for each record that is processed.

**False:**

(Default) Generate one metadata file all records that are processed.

# 21.106. PrinterProtocol

A `string` value that represents the printer protocol to use for printing a PDF document. `PrinterProtocol` variables are used to configure the sendToPrinter operation of the Output service.

For information about data that can be accessed using Xpath Expressions, see Dataitems.

For information about configuring default properties, see Datatypespecific settings.

## Data items

`PrinterProtocol` data values use one of these string values:

**CUPS:**

Common Unix Printing System protocol. An indirect method to access the print server.

**DirectIP:**

A direct method using internet protocols to access the print server that uses the default port 9100.

**LPD:**

Line Printer Daemon protocol. An indirect method to access the specified printer by name through the print server.

**SharedPrinter:**

A direct method to access the printer directly by name.

**CIFS:**

A direct method using the Common Internet File System printing protocol to access a printer or print server.

### Datatype specific settings

Property for specifying the default printer protocol to use.

**Default Value**

Sets the printer protocol to use. Select one of these values:

**CUPS:**

Common Unix Printing System protocol. An indirect method to access the print server.

**DirectIP:**

A direct method using internet protocols to access the print server that uses the default port 9100.

**LPD:**

Line Printer Daemon protocol. An indirect method to access the specified printer by name through the print server.

**SharedPrinter:**

A direct method to access the printer directly by name.

**CIFS:**

A direct method using the Common Internet File System printing protocol to access a printer or print server.

## 21.107. PrintFormat

A `string` value that represents the page description format to use when generating the output. The Output service supports these page description languages.

- PCL (Printer Command Language)
- ZPL (Zebra Programming language)
- DPL (Datamax Printer Language)
- IPL (Intermec Printer Language)
- TPCL (TEC Printer Command Language)
- PostScript (Adobe Page description language)

`PrintFormat` variables are used to configure the Print Format property of the generatePrintedOutput and generatePrintedOutput(deprecated) operations of the Output service.

For information about data that can be accessed using Xpath Expressions, see Dataitems.

For information about configuring default properties, see Datatypespecific settings.

## Data items

`PrintFormat` data values store one of these string values:

**PCL:**

Use the default XDC file for PCL or specify a custom XDC file for PCL.

**PostScript:**

Use the default XDC file or specify a custom XDC for PostScript.

**ZPL:**

Use the default XDC file for ZPL or specify a custom XDC file for ZPL.

**GenericColorPCL:**

Use a generic color PCL (5c).

**GenericPSLevel3:**

Use generic postscript level 3.

**ZPL3000DPI:**

Use ZPL 300 DPI.

**ZPL6000DPI:**

Use ZPL 600 DPI.

**IPL:**

Use custom IPL.

**IPL3000DPI:**

Use IPL 300 DPI.

**IPL4000DPI:**

Use IPL 400 DPI.

**TPCL:**

Use the default XDC file for TPCL or specify a custom XDC file for TPCL.

**TPCL305DPI:**

Use TPCL 300 DPI.

**TPCL600DPI:**

Use TPCL 600 DPI.

**DPL:**

Use the default XDC file for DPL or specify a custom XDC file DPL.

**DPL300DPI:**

Use DPL 300 DPI.

**DPL406DPI:**

Use DPL 400 DPI.

**DPL600DPI:**

Use DPL 600 DPI.

## Datatype specific settings

Property for specifying the print format to use.

### Default Value

The default print format to use. Select one of these values:

**Custom PCL:**

Use the default XDC file for PCL or specify a custom XDC file.

**Custom PostScript:**

Use the default XDC file for PostScript or specify a custom XDC for PostScript.

**Custom ZPL:**

Use the default XDC file for ZPL or specify a custom XDC file for ZPL.

**Generic Color PCL (5c):**

Use a generic color PCL (5c).

**Generic PostScript Level3:**

Use generic PostScript level 3.

**ZPL 300 DPI:**

Use ZPL 300 DPI.

**ZPL 600 DPI:**

Use ZPL 600 DPI.

**Custom IPL:**

Use custom IPL.

**IPL 300 DPI:**

Use IPL 300 DPI.

**IPL 400 DPI:**

Use IPL 400 DPI.

**Custom TPCL:**

Use the default XDC file for TPCL or specify a custom XDC file for TPCL.

**TPCL 305 DPI:**

Use TPCL 300 DPI.

**TPCL 600 DPI:**

Use TPCL 600 DPI.

**Custom DPL:**

Use the default XDC file for DPL or specify a custom XDC file DPL.

**DPL 300 DPI:**

Use DPL 300 DPI.

**DPL 406 DPI:**

Use DPL 400 DPI.

**DPL 600 DPI:**

Use DPL 600 DPI

## 21.108. PSLevel

A `string` value that can be used as the value for the PostScript Level property of the toPS(deprecated) operation that the Convert PDF service provides. Valid values are `LEVEL_2` and `LEVEL_3`.

## 21.109. ReadPermissionsResult

A complex data type that stores the results of the readPermissions operation that the Document Management service provides.

For information about data that can be accessed using Xpath Expressions, see Dataitems.

### Data items

The data items that the `ReadPermissionsResult` variables contain.

#### accessPermissions

A `list` of `ContentAccessPermission` values that contains the permissions that are set for the node.

#### inheritParentPermissions

A `boolean` value that specifies whether the permission is inherited from the parent node.

## 21.110. ReaderExtensionsOptionSpec

A complex data type used by the ApplyUsage Rights operation of the Acrobat Reader DC extensions service. This data type represents individual usage rights that you can apply to a PDF document and the message that appears when the rights-enabled PDF document is opened in Adobe Reader.

For information about data that can be accessed using Xpath Expressions, see Dataitems.

For information about configuring default properties, see Datatypespecific settings.

### Data items

The data items that `ReaderExtensionsOptionSpec` variables contain.

#### message

A `string` value that contains the message that appears when the right-enabled PDF document is opened in Adobe Reader.

#### modeFinal

A `boolean` value that specifies whether mode is set to final. The default value of `true` means that mode is set to final and the count of the credential used to apply usage rights to a PDF document is incremented. A value of `false` means that mode is not set to final.

#### usageRights

A `UsageRights` value that is used to specify usage rights when the user works with the PDF document in Adobe Reader. UsageRights can be one of these values:

**enabledBarcodeDecoding:**

A `boolean` value that indicates whether the PDF document supports barcoded forms decoding within Adobe Reader.

**enabledComments:**

A `boolean` value that indicates whether the user can create and modify document annotations such as comments within Adobe Reader.

**enabledCommentsOnline:**

A `boolean` value that indicates whether the user can upload and download annotations such as comments to and from an online document review and comment server.

**enabledDigitalSignatures:**

A `boolean` value that indicates whether the user can digitally sign and save PDF documents and clear digital signatures within Adobe Reader.

**enabledDynamicFormFields:**

A `boolean` value that indicates whether the user can add, change, or delete fields and field properties on the PDF form within Adobe Reader.

**enabledDynamicFormPages:**

A `boolean` value that indicates whether the user can create pages from the form template for forms created in Acrobat.

**enabledEmbeddedFiles:**

A `boolean` value that indicates whether the user can add, remove, modify, or export files attached to a PDF document within Adobe Reader.

**enabledFormDataImportExport:**

A `boolean` value that indicates whether the user can add, remove, modify, or export files attached to a PDF document within Adobe Reader.

**enabledFormFillIn:**

A `boolean` value that indicates whether the user can fill form fields and save the PDF document locally from within Adobe Reader.

**enabledOnlineForms:**

A `boolean` value that indicates whether the user can access the database or call the web service that is defined within the PDF document.

**enabledSubmitStandalone:**

A `boolean` value that indicates whether the user can submit data to a server by email or offline.

## Datatype specific settings

Properties for configuring usage rights for a PDF document.

**Basic Form Fill-in**

Determines whether basic form fill-in capabilities are allowed in the PDF document. When selected, basic form fill-in is allowed.

**Import and Export Form Data**

Determines whether form data can be imported or exported from a file. When selected, importing and exporting form data is permitted.

**Submit Outside Web Browser**

Determines whether a PDF document can be submitted outside a web browser. When selected, a user is permitted to submit a PDF document.

**Database and Web Service Connectivity**

Determines whether a PDF document can be used as an online form. When selected, a user is permitted to use a PDF document as an online form.

**Add, Delete, and Change Form Fields**

Determines whether existing filled-in form fields can be edited in the PDF document. When selected, existing form fields can be edited in a PDF document.

**Create Pages From Templates**

Determines whether adding new pages from existing templates in the PDF document is permitted. When selected, a user is permitted to add new pages from existing templates.

**2D Barcode Decoding**

Determines whether two-dimensional barcode decoding is permitted in the PDF document. When selected, two-dimensional barcode decoding is permitted.

**Digital Signatures**

Determines whether adding digital signatures to the PDF document is permitted. When selected, a user can add digital signature to a PDF document.

**Commenting**

Determines whether offline commenting of the PDF document is permitted. When selected, offline commenting is permitted.

**Online Commenting**

Determines whether online commenting of the PDF document is permitted. When selected, online commenting is permitted.

**Embedded File Attachments**

Determines whether to permit adding attachments to the PDF document. When selected, attachments can be added to a PDF document.

**Draft**

Determines whether the user is allowed to save the PDF document as a draft copy. When selected, a PDF document can be saved as a draft copy.

**Reader Message**

A message that represents the text displayed within Adobe Reader to inform users that the PDF document contains usage rights.

# 21.111. Reason

A `string` value that specifies the reason why a Rights Management license was revoked. This value is used by the Revokelicense operation in the Rights Management service.

These values are valid:

**DOCUMENT_TERMINATED:**

PDF document was revoked because it no longer exists. This value is the default value.

**DOCUMENT_REVISED:**

PDF document was revoked because it was revised.

**GENERAL_MESSAGE:**

An error occurred while processing the PDF document.

# 21.112. ReceiveProtocolEnum

A `string` value that represents the protocol to use for exchanging information with the POP3 or IMAP server. The value is used by the Receive operation in the Email service. The valid values are `pop3` and `imap`. The value you specify depends on the type of email server that you are using.

For information about configuring default properties, see Datatypespecific settings.

## Datatype specific settings

A list of protocols that can be used for exchanging information with the POP3 and IMAP servers.

# 21.113. RedactionOptionSpec

A complex data type used by the RedactPDF operation of the PDF Utilities service. This data type represents options for the redaction operation.

For information about data that can be accessed using Xpath Expressions, see Dataitems.

For information about configuring default properties, see Datatypespecific settings.

## Data items

Data itmes that the `RedactionOptionSpec` object contains.

### redactXObjectReferences

A `boolean` value that specifies whether to redact all references of the same XObject.

### redactWholeImageForUnsupportedFilter

A `boolean` value that specifies whether to redact the whole image when only parts of the image are marked for redaction but the image format is not supported by the image filters.

# 21.114. RedactionResult

A complex data type that represents the result of the RedactPDF operation of the PDF Utilities service.

For information about data that can be accessed using Xpath Expressions, see Dataitems.

For information about configuring default properties, see Datatypespecific settings.

## Data items

Data itmes that the `RedactionResult` object contains.

### document

A `document` value that represents the result of the redaction operation.

### redactionStatus

A `boolean` value that specifies whether the redaction operation was successful.

# 21.115. RelationInfo

A complex data type that stores the relationship information for items in an IBM Content Manager repository.

For information about data that can be accessed using Xpath Expressions, see Dataitems.

### Data items

The data items that `RelationInfo` variables contain.

**relationDesc**

A `string` value that represents the relationship description.

**relationType**

A `string` value that represents the type of relation.

**sourceDocInfo**

A `DocInfo` value that represents the metadata of the source item.

**targetDocInfo**

A `DocInfo` value that represents the metadata of the target item.

## 21.116. ReminderTO

Internal use only.

## 21.117. RenderAtClient

A `string` value that specifies whether the form is rendered on the client application. `RenderAtClient` values are used to configure the Render At Client property for renderPDFForm operation in the Forms service. the These string values are valid:

The Render At setting version on the server to open PDF Forms rendered by the Forms service. The Render At setting is configured in administration console. (See AEM Forms administration help.)

For information about data that can be accessed using Xpath Expressions, see Dataitems.

### Data items

`RenderAt` data values store one of these string values:

**Auto:**

The Forms service determines the form rendition based on the setting in the form design.

**Yes:**

A dynamic PDF form is generated and rendering occurs in Acrobat. Rendering occurs only on Acrobat 7.0 or later.

**No:**

A static PDF form is generated. No rendering on the client occurs.

# 21.118. RenderOptionsSpec-FormsService

A complex data type that represents run-time options that control how the Forms service handles form submissions. `RenderOptionsSpec-FormsService` variables are used to configure the Submission Options property in the processFormSubmission operation of the Forms service.

For information about data that can be accessed using Xpath Expressions, see Dataitems.

For information about configuring default properties, see Datatypespecific settings.

## Data items

The data items that `RenderOptionsSpec-FormsService` variables contain.

### acrobatVersion

An `acrobatVersion` value that represents the PDF version required for a client.

### cacheEnabled

A `boolean` value that indicates whether the Forms service caches a PDF form to improve performance. When caching is used, each form is cached after it is generated for the first time. On a subsequent render, if the cached form is newer than the form design's timestamp, the form is retrieved from the cache. A value of `true` indicates that the form is cached and `false` indicates it is not cached.

### CB

A `boolean` value that determines whether the generation of HTML-template initiating content, or guided forms content. `True` indicates to generate the HTML-template initiating content and `False` indicates to generate guided forms content.

### charset

A `string` value that represents the character set used to encode the output byte stream. The Forms service supports character encoding values defined by the `java.nio.charset` package for HTML transformations. For a complete list of these values, see http://java.sun.com/j2se/1.5.0/docs/guide/intl/locale.doc.html.

### clientCache

A `boolean` value that determines whether the form is cached in the client web browser cache. Only forms that are rendered as interactive PDF forms can be stored in the client web browser cache.

When client caching is used and subsequent requests for the PDF form are made, a timestamp located in the cached PDF form is compared with the timestamp of the PDF form that is generated by the Forms service and stored in the server cache. If the timestamps are the same, the PDF form is retrieved from the client cache. Retrieving PDF forms from the client cache results in reduced bandwidth usage and improves performance because the Forms service does not have to redeliver the same content to the client web browser.

A value of `true` indicates that client caching is used and `false` indicates client caching is not used.

### clientFrame

A `string` value that represents the client frame to use for HTML transformations. This value is not applicable for PDF forms.

### debugEnabled

A `boolean` value that determines whether debug-level logging is performed. Debug-level logging provides more information in the J2EE application server's log file. A value of `true` indicates that debug-level logging is performed, and `false` indicates that the default level of logging is performed.

### digSigCSSURI

A `string` value that represents the URI of a custom style sheet for a digital signature user interface in HTML forms.

### exportDataFormat

A `string` value that indicates the data format that the Forms service uses to export data to a client application. These values are valid:

**XDP:**

Returns the data with XDP packaging.

**XDPtoXFAData:**

Returns the data with XDP packaging and data in the `xfa:datasets` packet (xfdf is converted to xfa data).

**XDPDataOnly:**

XDP format but without an embedded PDF or XFA template packet.

**XMLData:**

Returns the data without XDP or xfa:datasets packaging.

**Auto:**

Automatically determines the format based on the submitted data:

- If XDP data is submitted, then XFA data is returned.
- If XML data is submitted, then XML data is returned.
- If URL-encoded data is submitted, then XMLdata is returned.

**fontMapURI**

A `string` value that represents the URI of the font mapping file. Font mapping defines which fonts are used in a form in place of a specified font that is not available. If the root of the URI is not specified, the file is assumed to reside in the EAR file.

**formModel**

A `FormModel` value that represents the location where the form is processed.

**guideAccessible**

A `boolean` value that indicates whether the form guide is accessible. A value of `true` indicates the form guide is created to be accessible and `false` indicates that it is not.

**guideCBURL**

A `string` value that represents the URL for a custom override implementation of the callback servlet.

**guideName**

A `string` value that represents the name of the guide within the generated form guide.

**guidePDF**

A `boolean` value that determines whether the output is rendered in the form guide and PDF form. A value of `true` indicates that both the form guide and PDF form are rendered and `false` indicates that only the form guide is rendered.

**guideRSL**

A `boolean` value that determines whether to use run-time shared libraries when compiling a form guide. A value of `true` indicates that shared libraries are used and `false` indicates they are not used.

**guideStyle**

A `string` value that represents the name of the style sheet that the form guide uses. The style sheet is also embedded in the form guide. The application container contains the style sheet, or the style sheet can be referenced.

### guideSubmitAll

A `boolean` value that determines whether the form guide submits all data including hidden panels. A value of `true` indicates that all data is submitted and `false` indicates that all data is submitted except for data from hidden panels.

### ifModifiedSince

For internal use only.

### imageURL

A `string` value that represents the URL that receives image requests from the target device. This URL is inserted into the transformed output to facilitate retrieval of embedded images, such as images embedded within an XDP form.

This value is required only when authentication is required to retrieve the images.

If the value is not an absolute URL, the value of ApplicationWebRoot is combined with this value to construct an absolute URL. (See URLSpec.)

### injectFormBridge

A `boolean` value that indicates whether the Forms service inserts special JavaScript code into a PDF form. The code performs operations on a PDF form when rendering guided forms. A value of `true` indicates the Forms service inserts the JavaScript code and `false` indicates it does not.

### internalOptionMap

A `map` of `string` values that is for internal use only.

### linearizedPDF

A `boolean` value that indicates whether the Forms service produces a linearized PDF form (optimized for web applications). A linearized PDF document is organized to enable incremental access in a network environment. For example, a linearized PDF document can be displayed in a web browser before the entire PDF document is downloaded.

A value of `true` indicates the PDF form is linearized and `false` indicates it is not linearized.

### locale

A `string` value that indicates the locale that the Forms service uses to send validation messages to client devices, such as web browsers, when an HTML form is rendered. For a complete list of supported locale codes, see http://java.sun.com/j2se/1.5.0/docs/guide/intl/locale.doc.html.

### options

A `string` value that represents the options that are set.

### outputType

An `int` value that specifies how a form that is rendered as HTML is displayed. These values are valid:

**0:**

The form is rendered within `<HTML>` elements.

**1:**

The form is rendered within `<body>` elements.

### pageNumber

An `int` value that represents the initial page number to render in a multipage HTML form. This option is not applicable to PDF forms.

### PDF2XDP

A `boolean` value that determines whether the Forms service produces XDP output from submitted PDF content. By default, the Forms service returns the PDF content that is submitted.

If XDP output is produced, it contains the following elements:

- A PDF data (base64-encoded).
- A datasets packet that contains form data if the PDF content is XFA-based.
- An XFDF packet that stores annotations. If the PDF content is an Acrobat PDF form, the XFDF also contains elements that represent form fields.

### PDFVersion

A `string` value that represents the PDF version of the PDF form that is rendered. These string values are valid:

**PDFVersion_1_5:**

Represents PDF version 1.5.

**PDFVersion_1_6:**

Represents PDF version 1.6.

**PDFVersion_7:**

Represents PDF version 1.7.

**NOTE:** Acrobat 6.0 supports PDF version 1.5. Acrobat 7.0 and Acrobat 8.0 support PDF version 1.6.

### propertyMap

A `map` of `string` values that is for internal use only.

### re2DBarcode

A `boolean` value that determines whether the Forms service renders forms that contain two-dimensional (2D) barcodes and enables a user to fill the form using Adobe Reader. A value of `true` indicates that 2D barcodes can be rendered and `false` indicates that 2D barcodes cannot be rendered.

When a user fills an interactive form that contains a barcode, the barcode is updated automatically to encode the user-supplied data.

When you set this property value to `true`, the Acrobat Reader DC extensions service must be available. You must also set values for both the reCredentialAlias and reCredentialPassword properties.

### reCommenting

A `boolean` value that determines whether the Forms service renders forms that allow users to add comments to the PDF form using Adobe Reader. A value of `true` indicates the service does render these forms and `false` indicates these forms cannot be rendered.

When you set this property value to `true`, the Acrobat Reader DC extensions service must be available. You must also set values for both the reCredentialAlias and reCredentialPassword properties.

### reCreatePages

A `boolean` value that indicates whether the Forms service renders PDF forms that allow users to dynamically add pages within Adobe Reader. A value of `true` indicates that the service does render these forms and a value of `false` indicates it does not.

When you set this property value to `true`, the Acrobat Reader DC extensions service must be available. You must also set values for both the reCredentialAlias and reCredentialPassword properties

### reCredentialAlias

A `string` value that represents the alias of the credential that grants Acrobat Reader DC extensions usage rights.

### reCredentialPassword

A `string` value that represents the password for the alias that is specified for the reCredentialAlias property.

### reDigSig

A `boolean` value that determines whether the Forms service renders PDF forms that allow users to digitally sign the form using Adobe Reader. A value of `true` indicates that these forms can be rendered and a value of `false` indicates these forms cannot be rendered.

When you set this property value to `true`, the Acrobat Reader DC extensions service must be available. You must also set values for both the reCredentialAlias and reCredentialPassword properties

### reEmbeddedAttachments

A `boolean` value that determines whether the Forms service renders PDF forms that allow users to add attachments to the form using Adobe Reader. A value of `true` indicates that these forms can be rendered and a value of `false` indicates these forms cannot be rendered.

When you set this property value to `true`, the Acrobat Reader DC extensions service must be available and you need to set values for both the reCredentialAlias and reCredentialPassword properties.

### reExpImp

A `boolean` value that determines whether the Forms service renders PDF forms that allow users to import and export form data using Adobe Reader. A value of `true` indicates that these forms can be rendered and a value of `false` indicates these forms cannot be rendered.

When you set this property value to `true`, the Acrobat Reader DC extensions service must be available. You must also set values for both the reCredentialAlias and reCredentialPassword properties.

### reFillIn

A `boolean` value that determines whether the Forms service renders PDF forms that allow users to fill form fields and save the PDF form using Adobe Reader. A value of `true` indicates that these forms can be rendered and a value of `false` indicates these forms cannot be rendered.

When you set this property value to `true`, the Acrobat Reader DC extensions service must be available. You must also set values for both the reCredentialAlias and reCredentialPassword properties.

### reFormFieldMod

A `boolean` value that determines whether the Forms service renders PDF forms that allow users to modify form fields and save the PDF form using Adobe Reader. A value of `true` indicates that these forms can be rendered and a value of `false` indicates these forms cannot be rendered.

When you set this property value to `true`, the Acrobat Reader DC extensions service must be available.You must also set values for both the reCredentialAlias and reCredentialPassword properties.

### renderAtClient

A `RenderAtClient` value that determines whether forms are rendered to PDF content on the server or on the client (Acrobat). When the value is `auto` and the acrobatVersion property is `Acrobat_7_0_5` or `Acrobat_8`, a form is rendered to a client once, which improves the performance of the Forms service. The rendered form behaves as a non-interactive form. (See acrobatVersion.)

### reOnlineCommenting

A `boolean` value that determines whether the Forms service renders the PDF form to allow users to add comments using Adobe Reader while online. A value of `true` indicates that these forms can be rendered and a value of `false` indicates these forms cannot be rendered.

When you set this property value to `true`, the Acrobat Reader DC extensions service must be available. You must also set values for both the reCredentialAlias and reCredentialPassword properties.

### reOnlineForms

A `boolean` value that determines whether the Forms service renders PDF forms that allow users to interact with forms while online using Adobe Reader. A value of `true` indicates that these forms can be rendered and a value of `false` indicates these forms cannot be rendered.

When you set this property value to `true`, the Acrobat Reader DC extensions service must be available. You must also set values for both the reCredentialAlias and reCredentialPassword properties.

### reReaderMessage

A `string` value that represents the message that appears in Adobe Reader that informs users that the form contains Acrobat Reader DC extensions usage rights.

### reStandaloneSubmit

A `boolean` value that determines whether the Forms service should render PDF forms that allow users to submit information from within Adobe Reader. A value of `true` indicates that these forms can be rendered and a value of `false` indicates these forms cannot be rendered.

When you set this property value to `true`, the Acrobat Reader DC extensions service must be available and the reCredentialAlias and reCredentialPassword properties are set to True.

### rootLocale

A `string` value that is for internal use only.

### seedPDF

A `string` value that represents the name and location of the PDF document to use as the shell PDF (seedPDF) document.

A shell PDF contains only an XFA stream, font and image resources, and one page that is either blank or contains a warning that the document must be opened using Acrobat 7 or later or Adobe Reader 7. The shell PDF is used with PDF transformation to optimize delivery of PDFForm transformations only.

### serviceId

A `string` value that is for internal use only.

### standAlone

A `boolean` value that specifies whether the form is rendered without state information. This value is useful when the rendering of an interactive form occurs on the server instead of on the client, and the form contains JavaScript code that is executed.

A value of `true` indicates that the code that the form contains runs on the client with no interaction with the server. A value of `false` indicates that state information is used to render an interactive form to an end user who then enters information into the form and submits the form back to the Forms service. The Forms service then performs a calculation operation and renders the form back to the user with the results displayed in the form.

### taggedPDF

A `boolean` value that determines whether the Forms service produces a tagged PDF form. A value of `true` indicates a tagged PDF is produced and a value of `false` indicates the PDF is not tagged.

Tags in a PDF form define a set of standard structure types and attributes. These standard types and attributes allow page content (text, graphics, and images) to be extracted and reused for other purposes. It is intended for use by tools that perform the following types of operations:

- Simple extraction of text and graphics for pasting into other applications.
- Automatic reflow of text and associated graphics to fit a page of a different size than was assumed for the original layout.
- Processing text for such purposes as searching, indexing, and spell-checking.
- Conversion to other common file formats (such as HTML, XML, and RTF) with document structure and basic styling information preserved.
- Making content accessible by screen reader software.

Tagged PDF files are not supported in PDF versions earlier than version 1.4.

### toolbarMenu

A `HTMLToolbar` value that represents the appearance of an HTML toolbar that is rendered with an HTML form.

### validationBorder

A `string` value that represents the validation border size, in pixels, when using frames in an HTML form.

### validationReporting

An `int` value that represents the position within the frame where validation reporting is displayed if the ValidationUI option is 0 (list). This value specifies whether the list appears within a frame, and the location of the list relative to the form. This property is valid for forms that are rendered as HTML. These values are valid:

**0:**

The error is specified on the left of the form in a separate frame.

**1:**

The error is specified on the right of the form in a separate frame.

**2:**

The error is specified at the top of the form in a separate frame.

**3:**

The error is specified at the bottom of the form in a separate frame.

**4:**

The error is specified on the left of the page without a frame.

**5:**

The error is specified on the right of the page without a frame.

**6:**

The error is specified at the top of the page without a frame.

**7:**

The error is specified at the bottom of the page without a frame.

**8:**

No error reporting is provided.

**9:**

The error is logged but is not provided to the user interface.

**10:**

The error is not displayed if a form is within the frame.

**validationUI**

An `int` value that represents how errors are displayed. These values are valid:

**0:**

list

**1:**

message box

**XCI**

(Optional) A `string` value that represents XCI run-time options that are executed for renderForm and processFormSubmission operations. These XCI options override configuration values that are located within the XCI configuration file that the Forms service uses. The value has the following format:

```
[XPath-expression]=[value];[XPath-expression]=[value];...
```

- `[path-expression]` is an XPath expression that resolves to the element in the XCI file to update. The XPath expression is relative to the root element of the XCI file. Only simple XPath expressions that contain "/" and "@" characters are supported.
- `[value]` is the value to assign to the path expression. If the value is omitted, the element is cleared.

The following example changes the default typeface to Myriad® Pro:

```
present/pdf/fontInfo/defaultTypeface=MyriadPro
```

Setting XCI options can affect the resulting PDF form. For example, consider the following XCI options:

```
present/pdf/fontInfo/embed=1&PDFVersion=1.6
present/pdf/fontInfo/embed=1
```

The size of the resulting files is different. When you set the PDF version to 1.6 the Forms service embeds the fonts as a subset. When you do not set the PDF version, the Forms service fully embeds the font. The PDF that contains the fully embedded font is at least twice as large as the file that has the fonts embedded as subset. Specify the PDF version to embed the font as a subset.

The following XCI elements can be configured:

- acrobat/acrobat7/dynamicRender
- data/outputXSL/uri (child of outputXSL element)
- data/range
- data/record
- data/startNode
- data/xsl/debug/uri (child of debug element; descendant of data element)
- data/xsl/uri (child of xsl element; descendant of data element)
- destination
- fontInfo/defaultTypeface
- locale
- pdf/compression/compressLogicalStructure
- pdf/compression/level
- pdf/compression/type
- pdf/encryption/encrypt
- pdf/encryption/encryptionLevel
- pdf/encryption/masterPassword
- pdf/encryption/permissions/accessibleContent
- pdf/encryption/permissions/contentCopy
- pdf/encryption/permissions/documentAssembly
- pdf/encryption/permissions/formFieldFilling
- pdf/encryption/permissions/modifyAnnots
- pdf/encryption/permissions/plaintextMetadata
- pdf/encryption/permissions/print
- pdf/encryption/permissions/printHighQuality
- pdf/encryption/permissions/change
- pdf/encryption/userPassword
- pdf/fontInfo/embed
- pdf/fontInfo/encodingSupport
- pdf/fontInfo/map/equate
- pdf/fontInfo/subsetBelow

- pdf/interactive
- pdf/openAction/destination
- pdf/submitFormat
- pdf/tagged
- pdf/xdc/uri
- present/pdf/linearized
- present/pdf/renderPolicy
- temp/uri (descendant of pdf element)
- template/base (descendant of temp element)

**XCIURI**

A `string` value that represents the URI location of the XCI file to use for rendering. If the root is not specified, the file is assumed to reside in the EAR file.

**XDCURI**

A `string` value that represents the URI location of an alternative XDC file to use for rendering. This property can be set to a valid URI, an absolute URI with protocols, or a file location. An alternative XDC file is typically used when the Forms service cannot locate an XDC file. For example, there is a network issue.

**XMLData**

A `boolean` value that indicates whether the Forms service produces the form's XML data based on its current processing state. A value of true indicates that the current processing state is used and false indicates that some other state is used.

## Datatype specific settings

Properties for configuring the rendering options when processing a submission to the Forms service.

**HTML Output Type**

Set whether to wrap the rendered page with Full HTML Tags or Body Tags. The default value is <Use Server Default>. Select one of these string values:

**<Use Server Default>:**

Use the default Output type setting configured on AEM Forms Server. The Output type setting is configured using Forms in Administration Console. (See AEM Forms administration help.)

**Full HTML tags:**

The rendered page is wrapped with `<HTML>` tags.

**Body Tags:**

The rendered page is wrapped with `<BODY>` tags.

### Character Set

Sets the character set used to encode the output byte stream. The default value is <Use Server Default>. Select either a character set or one of these values.

**<Use Server Default>:**

Use the Character Set setting configured on AEM Forms Server. The Character Set setting is configured using Forms in administration console. (See AEM Forms administration help.)

**<Use Custom Value>:**

Use a character set not available in list. After selecting this value, in the box beside the list, type the canonical name (Java.nio API) of the encoding set to use. For a complete list of character sets, see http://java.sun.com/j2se/1.5.0/docs/guide/intl/encoding.doc.html.

### Locale

Sets the language used to send validation messages to client devices, such as web browsers, when an HTML form is rendered. The default value is <Use Server Default>. Select either a language from the list or one of these values.

**<Use Server Default>:**

Use the Locale setting configured on AEM Forms Server. The Locale setting is configured using Forms in administration console. (See AEM Forms administration help.)

**<Use Custom Value>:**

Use a locale that is not available in the list. After selecting this value, in the text box beside the list, type the Locale ID of the locale code to use. For a complete list of supported locale codes, see http://java.sun.com/j2se/1.5.0/docs/guide/intl/locale.doc.html.

### Display Validation Messages

Set whether to display messages in the web page or as a separate message box. The default value is <Use Server Default>. Select one of these string values:

**<Use Server Default>:**

Use the Reporting setting configured on AEM Forms Server. The Reporting setting is configured using Forms in administration console. (See AEM Forms administration help.)

**In The Page:**

Specifies that errors are returned as a list within the web page.

**In The Message Box:**

Specifies that errors are returned in a message box.

### Validation Reporting

Sets a value to specify the position within the frame where validation reporting is displayed. This property is used when the Display Validation Messages option is set to In The Page. The default value is <Use Server Default>. Select one of these string values:

**<Use Server Default>:**

Use the Position setting configured on AEM Forms Server. The Position setting is configured using Forms in administration console. (See AEM Forms administration help.)

**Frame Left:**

The error is specified on the left of the form in a separate frame.

**Frame Right:**

The error is specified on the right of the form in a separate frame.

**Frame Top:**

The error is specified at the top of the form in a separate frame.

**Frame Bottom:**

The error is specified at the bottom of the form in a separate frame.

**No Frame Left:**

The error is specified on the left of the page without a frame.

**No Frame Right:**

The error is specified on the right of the page without a frame.

**No Frame Top:**

The error is specified at the top of the page without a frame.

**No Frame Bottom:**

The error is specified at the bottom of the page without a frame.

**None:**

No error reporting is provided.

**No UI:**

The error is logged but is not provided to the user interface.

**No UI With Form:**

The error is not displayed if a form is within the frame.

### Validation Border

Type a value to specify the validation border size in pixels when using Frames in the Validation Reporting option. Use any integer value. When a value is not provided, the Border Size setting configured on AEM Forms Server is used. The Border Size setting is configured using Forms in administration console. (See AEM Forms administration help.)

### Populate XML Data

Sets whether the XML data is produced from the form based on the form's current processing state. The default value is False. Select one of the values:

**False:**

Specifies not to produce the XML data.

**True:**

Specifies to produce the XML data.

### PDF to XDP

Select a value to specify whether to generate XDP output from submitted PDF content. The default value is False. Select one of these values.

**True:**

Generate the XDP output from the PDF content.

**False:**

Do not generate XDP output.

### Export Data Format

Select a value to specify the format of the data that is exported to a client application. The default value is XDP. Select one of these string values:

**<Use Form Template Default>:**

Specifies to automatically determine the format based on the data that is submitted:

- If XDP data is submitted, then XFA data is exported.
- If XML data is submitted, then XML data is exported.
- If URL-encoded data is submitted, then XML data is exported.

**XDP:**

Export the data with XDP packaging.

**XDP to XFA Data:**

Export the data with XDP packaging and data in the xfa:datasets packet (xfdf is converted to xfa data).

**XDP Data Only:**

Export the data with XDP packaging but without an embedded PDF or XFA template packet.

**XML Data:**

Export the data without XDP or xfa:datasets packaging.

**XCI URI**

Type a value to specify the URI location of the XCI file to use for rendering. If the root is not specified, the file is assumed to reside in the location where the AEM Forms EAR files are deployed.

# 21.119. RenderOptionsSpec-OutputService

A complex data type that represents options to control how the Output service generates output, such as setting XCI options or caching form designs in order to improve performance. `RenderOptionsSpec-OutputService` variables are used to configure the Render Options property in the generatePDFOutput (deprecated) operation of the Output service.

For information about data that can be accessed using Xpath Expressions, see Dataitems.

For information about configuring default properties, see Datatypespecific settings.

## Data items

The data items that `RenderOptionsSpec-OutputService` variables contain.

**cacheEnabled**

A `boolean` value that determines whether the Output service should cache a form design to improve performance. A value of `true` indicates that caching is performed and `false` indicates that caching is not performed.

When caching is performed, a form design is cached the first time it is used to generate PDF document output. The next time the form design is used to generate PDF document output, the Output service determines whether the cached form design is newer than the non-cached form design by comparing timestamps. If the cached form design is newer, it is retrieved from the cache.

This option works only with the caching property of the form design. (See*Designer Help*.)

**debugEnabled**

A `boolean` value that determines whether debug-level logging is performed. A value of `true` indicates that debug-level logging is performed, and `false` indicates that the default level of logging is performed.

### linearizedPDF

A `boolean` value that indicates whether the Output service produces a linearized PDF form (optimized for web applications). A value of `true` indicates the PDF form is linearized and `false` indicates it is not linearized.

A linearized PDF document is organized to enable incremental access for network applications. For example, a linearized PDF document can be displayed in a web browser before the entire PDF document is downloaded.

### options

A `string` representation of the options that are set.

### PDFAAmendment

A `string` value that represents the PDF/A document amendment identifier. This value also specifies the year that is separated from the amendment identifier value by a colon.

For information about the PDF/A document amendment value, see *Document management - Electronic document file format for long-term preservation* specification (ISO-19005-1:2005).

### PDFAConformance

A `PDFAConformance` value that represents the PDF/A conformance level to use, as specified in the PDF/A-1 ISO specification. The conformance level refers to how a PDF document adheres to requirements that specify how long-term electronic documents are preserved.

For information about level A and B conformance, see *Document management - Electronic document file format for long-term preservation* specification (ISO-19005-1:2005).

### PDFARevisionNumber

An `PDFARevisionNumber` value that specifies the revision number of a PDF/A document. (See *Document management - Electronic document file format for long-term preservation* specification (ISO-19005-1:2005)).

### pdfVersion

A `string` value that represents the PDF version value that represents the PDF version to output (when formPreference is PDFForm or PDFMerge).

These values are valid:

- 1.5
- 1.6
- /A

Acrobat 6 supports PDF version 1.5. Acrobat 7 and Acrobat 8 supports PDF version 1.6. The value /A specifies a standard for archiving electronic documents (see *Document management - Electronic document file format for long-term preservation* specification (ISO-19005-1:2005)). This value embeds all the fonts and turns off compression. A PDF/A document is much larger than a standard PDF document.

### renderAtClient

A `string` value that determines whether PDF content is rendered on the client application by using Acrobat 7.0 or later or on the server.

### retainSignatureField

A `RetainSignatureField` value that specifies whether signature fields remain interactive after the PDF document is flattened.

### taggedPDF

A `boolean` value that determines whether the Output service produces a tagged PDF form. A value of `true` indicates a tagged PDF is produced and a value of `false` indicates the PDF is not tagged.

Tags in a PDF form define a set of standard structure types and attributes that allow page content (text, graphics, and images) to be extracted and reused for other purposes. It is intended for use by tools that perform the following types of operations:

- Simple extraction of text and graphics for pasting into other applications.
- Automatic reflow of text and associated graphics to fit a page of a different size than was assumed for the original layout.
- Processing text for such purposes as searching, indexing, and spell-checking.
- Conversion to other common file formats (such as HTML, XML, and RTF) with document structure and basic styling information preserved.
- Making content accessible by screen reader software.

Tagged PDF files are not supported in PDF versions earlier than version 1.4.

## Datatype specific settings

Properties for setting options for generating a PDF document.

### Acrobat Version

Sets a value to specify the Acrobat version that is required to view the PDF form that is rendered. Select one of these values:

**Acrobat and Adobe Reader 6 or later:**

PDF Version 1.5 is used to generate the PDF document.

**Acrobat and Adobe Reader 7.0 or later:**

PDF Version 1.6 is used to generate the PDF document.

**Acrobat and Adobe Reader 7.0.5 or later:**

PDF Version 1.65 is used to generate the PDF document.

**Acrobat and Adobe Reader 8 or later:**

PDF Version 1.7 is used to generate the PDF document.

**Acrobat and Adobe Reader 8.1 or later:**

PDF Version 1.7-ADBE-1 is used to generate the PDF document.

**Acrobat and Adobe Reader 9 or later:**

PDF Version 1.7-ADBE-3 is used to generate the PDF document.

**Auto:**

(Default) The Target Version setting in the form design determines the minimum version of Acrobat or Adobe Reader. In addition, the form design determines the PDF version.

**Tagged PDF**

Sets whether to create a tagged Adobe PDF form. A tagged PDF form defines a set of standard structure types and attributes that support the extraction of page content and reuse for other purposes. It is intended for use by client applications that perform the following types of operations:

- Simple extraction of text and graphics for pasting into other applications.
  - Automatic reflow of text and associated graphics to fit a page of a different size than was assumed for the original layout.
  - Processing text for such purposes as searching, indexing, and spell-checking.
- Conversion to other common file formats (such as HTML, XML, and RTF) with document structure and basic styling information preserved.
- Making content accessible by screen reader software.

Select one of these values:

**False:**

(Default) Do not render a tagged PDF form.

**True:**

Render a tagged PDF form. This value is not valid for PDF/A output because PDF/A 1A is always tagged and PDF/A 1B is never tagged.

**Linearized PDF**

Sets whether to render a linearized PDF form. A linearized PDF form is organized so that it supports incremental access in a network environment. For example, a linearized PDF can be displayed in a web browser before the entire PDF document is downloaded. Select one of these values:

**False:**

(Default) Do not render a linearized PDF form. It is best to use this option for non-web applications.

**True:**

Render a linearized PDF form. It is best to use this option for optimized web applications.

**Render At Client**

Sets whether to enable the delivery of PDF content by using the client-side rendering capability of Acrobat 7.0 or Adobe Reader 7.0 and later. Client-side rendering improves the performance of the Forms service. Select one of these values:

**Auto:**

(Default) The Output service determines the form rendition based on the setting in the form design.

**Yes:**

A dynamic PDF form is generated and rendering occurs in Acrobat. Rendering of a dynamic form occurs only in Acrobat 7.0 or later. For earlier version of Acrobat, no rendering occurs.

**No:**

A static PDF form is generated. No rendering on the client occurs.

**Retain Signature Field**

Sets whether Signature Fields are retained in the output. Select one of these values:

**None:**

Signed field and unsigned signature fields become non-interactive when the PDF file is flattened.

**All:**

Signed fields and unsigned signature fields are remain interactive when the PDF file is flattened.

**Signed Signature Fields:**

Unsigned signature fields become non-interactive. Signed signature fields remain interactive when the PDF file is flattened.

**Unsigned Signature Field Only:**

Signed signature fields become non-interactive. Unsigned signature fields are remain interactive when the PDF file is flattened.

**Debug Enabled**

Sets whether debugging-level logging is turned on. Debug-level logging provides more information in the J2EE application server's log file for debugging. Select one of these values:

**True:**

Debug-level logging is performed,

**False:**

(Default) Default level of logging is performed.

**PDF/A Revision Number**

Sets version for the PDF/A revision number. The default is Revision_1.

**PDF/A Conformance**

Sets the conformance level with the PDF/A-1 specification to use for archival and long-term preservation of electronic documents. Select one of these values:

**A:**

(Default) Level A conformance specifies complete conformance with ISO 19005. The PDF file is generated using PDF 1.4 and all colors are converted to either CMYK or RGB. These PDF files can be opened in Acrobat and Adobe Reader 5.0 and later.

**B:**

Level B conformance specifies minimal compliance with ISO 19005. The PDF file is generated with all fonts embedded, the appropriate PDF bounding boxes specified, and colors as CMYK, spot colors, or both. Compliant files must contain information describing the printing condition they are prepared for. PDF files created with PDF/X-1a compliance can be opened in Acrobat 4.0 and Adobe Reader 4.0 and later.

# 21.120. RetainSignatureField

A `string` value that represents specifies whether signed and unsigned signature fields are kept when a PDF file is flattened. `RetainSignatureField` variables are used to configure the Retain Signature Field property in the generatePDFOutput operation of the Output service.

For information about data that can be accessed using Xpath Expressions, see Dataitems.

For information about configuring default properties, see Datatypespecific settings.

## Data items

`RetainSignatureField` data values store one of these string values:

**None:**

Signed field and unsigned signature fields become non-interactive when the PDF file is flattened.

**All:**Signed fields and unsigned signature fields are remain interactive when the PDF file is flattened.

**Signed_Signature_Fields_Only:**

Unsigned signature fields become non-interactive. Signed signature fields remain interactive when the PDF file is flattened.

**Unsigned_Signature_Fields_Only:**

Signed signature fields become non-interactive. Unsigned signature fields are remain interactive when the PDF file is flattened.

### Datatype specific settings

Property for specifying how signature fields are handled during the flattening process of a PDF document.

#### Default value

Sets whether to make signature fields non-interactive in the rendered PDF document. Select one of these values:

**None:**

Signed field and unsigned signature fields become non-interactive when the PDF file is flattened.

**All:**

Signed fields and unsigned signature fields are remain interactive when the PDF file is flattened.

**Signed Signature Fields:**

Unsigned signature fields become non-interactive. Signed signature fields become interactive when the PDF file is flattened.

**Unsigned Signature Field Only:**

Signed signature fields become non-interactive. Unsigned signature fields are remain interactive when the PDF file is flattened.

## 21.121. ReviewContextTO

A complex data type that represents the unique identifier and other information for an instance of a review. Used with the Review, Commenting, and Approval building block, which is installed with the Managed Review & Approval Solution Accelerator.

For information about data that can be accessed using XPath Expressions, see Dataitems.

### Data items

The data items that ReviewContextTO values contain.

#### additionalMetadata

A `string` value that represents additional information that initiators can provide when they create a review.

**arsProcessName**

A `string` value that represents the process that attaches an approval routing slip to the review document.

**auditLevel**

A `string` value that represents the audit level of the review. For workflows that involve regulatory compliance.

**changeDescription**

A `string` value that represents the revision status of the review. Used if a review has been revised.

**commentServerPath**

A `string` value that represents the server location where inline comments are stored.

**commentVisibility**

A `string` value that represents the type of comment visibility associated with the review.

**complianceCode**

A `string` value that represents the legislation that this review complies with.

**currentRevision**

A `boolean` value that indicates whether this instance is the current revision of the review. If `true`, the review is the current revision.

**currentStage**

An `int` value that represents the current stage number of the review (starts with 1).

**customAttributes**

A `list` of CustomAttributeTO values that represent a list of custom attributes associated with the review.

**id**

Internal use only.

**initiator**

An InitiatorTO value that represents the initiator of the review.

**invocationId**

A `string` value that represents the unique identifier of the invocation used to start the review process.

**purpose**

A `string` value that represents the purpose of the review.

**reviewId**

A `string` value that represents the unique identifier of the review.

**reviewType**

A `string` value that represents the type of review. Valid values are REGULATED or AD-HOC.

**revision**

An `int` value that represents the current revision of the review.

**rtsProcessName**

A `string` value that represents the process that generates a review tracking sheet for the review.

**stageList**

A `list` of ReviewStageTO values that represent the stage values for the review.

**status**

A `string` value that represents the status of the review. The following values are valid:

- ONGOING
- COMPLETED
- EXPIRED
- REVOKED
- REVISED

**stp**

A `boolean` value that indicates whether straight-through processing (STP) is enabled. If `true`, STP is enabled.

**supportingDocumentList**

A `list` of DocumentTOvalues that represent the supporting documents associated with the review.

**templateAuthor**

A `string` value that represents the author of the template for the review.

**templateDescription**

A `string` value that represents a description of the template for the review.

### templateName

A `string` value that represents the name of the template for the review.

### title

A `string` value that represents the title of the review.

# 21.122. ReviewStageTO

A complex data type that represents a stage in a multi-staged review and approval process. Used with the Review, Commenting, and Approval building block, which is installed with the Managed Review & Approval Solution Accelerator.

For information about data that can be accessed using XPath Expressions, see Dataitems.

## Data items

The data items that ReviewStageTO values contain.

### additionalMetadata

A `string` value that represents additional information associated with the stage.

### assignTaskToInitiatorProcess

A `string` value that represents the process that assigns the task to the initiator.

### assignTaskToParticipantProcess

A `string` value that represents the process that assigns the task to a stage participant.

### disposition

A `string` value that represents a custom status for the stage.

### duration

An `int` value that represents the duration of the stage.

### durationUnit

A `string` value that represents the unit of measurement for the stage duration.

### endDate

A `date` value that specifies the end of the stage.

**id**

Internal use only.

**name**

A `string` value that represents the name of the stage.

**nonExpiringStage**

A `boolean` value that indicates whether this stage does not expire. If `true`, the stage is non-expiring.

**participants**

A `list` of ParticipantTO values that represent the stage participants.

**postProcessHookName**

A `string` value that represents the post-process hook for the stage.

**preProcessHookName**

A `string` value that represents the pre-process hook for the stage.

**reminders**

A `list` of ReminderTO values that represent reminders for the stage.

**reviewContext**

A ReviewContextTO value that the stage is associated with.

**signatureType**

A `string` value that represents a signature type for an approval stage.

**stageNo**

An `int` value that represents the stage's number. Starts with 1.

**startDate**

A `date` value that specifies the start of the stage.

**status**

A `string` value that represents the status of the stage. The following values are valid:

- PENDING
- ONGOING
- COMPLETED
- EXPIRED

- REVISED
- REVOKED
- APPROVED
- REJECTED

**taskType**

A `string` value that represents the type of task.

**type**

A `string` value that represents the stage type. The following values are valid:

- PARALLEL_REVIEW
- SEQUENTIAL_REVIEW
- PARALLEL_APPROVAL
- SEQUENTIAL_APPROVAL

**waitForExpiry**

A `boolean` value that indicates whether to wait for the stage to complete its duration regardless of the status of the participants. If `true`, the process does not proceed to the next stage until the duration is complete.

# 21.123. ReviewTemplateTO

A complex data type that represents a review template. Used with the Review, Commenting, and Approval building block, which is installed with the Managed Review & Approval Solution Accelerator.

For information about data that can be accessed using XPath Expressions, see Dataitems.

## Data items

The data items that ReviewTemplateTO values contain.

**active**

A `boolean` value that indicates whether the template is active. If `true`, the template is active.

**author**

A `string` value that represents the template's author.

**complianceCode**

A `string` value that represents the legislation that this review complies with.

**customAttributes**

A `list` of CustomAttributeTO values that represent custom attributes associated with the review template.

**description**

A `string` value that represents the description of the review template.

**id**

Internal use only.

**name**

A `string` value that represents the name of the template.

# 21.124. RevocationCheckStyle

A `string` value that represents the type of revocation-checking that the Verify PDFSignature and VerifyPDF Signature (deprecated) operations perform when verifying a signature in a PDF document. The default value is BestEffort. These string values are valid:

**AlwaysCheck:**

Checks for revocation of all certificates.

**BestEffort:**

Checks for revocation of all certificates when possible.

**CheckIfAvailable:**

Checks for revocation of all certificates only when revocation information is available.

**NoCheck:**

Does not check for revocation.

For information about configuring default properties, see Datatypespecific settings.

## Datatype specific settings

A list of revocation-checking styles to use to verify a signature in a PDF document. The default value is BestEffort. Select one of these values:

These string values are valid:

**AlwaysCheck:**

Checks for revocation of all certificates.

**BestEffort:**

Checks for revocation of all certificates when possible.

**CheckIfAvailable:**

Checks for revocation of all certificates only when revocation information is available.

**NoCheck:**

Does not check for revocation.

# 21.125. RevocationInformation

A complex data type that contains information about revoked certificates. `RevocationInformation` variables are members of `CertificateInformation` variables.

For information about data that can be accessed using Xpath Expressions, see Dataitems.

## Data items

The data items that `RevocationInformation` variables contain.

**data**

A `byte` value that represents the revocation identifier.

**source**

A `string` value that represents the source that was used to retrieve revocation information.

**status**

A `string` value that represents the status of the revocation for the certificate. These values are string valid:

**Unknown:**

The status could not be verified.

**Cache:**

The status of the revocation is cached on AEM Forms Server.

**Online:**

The status of the revocation is determined by accessing the network.

**Embedded:**

The status of the revocation is embedded from the certificate.

**DocumentSecurityStore:**

The status of the revocation is retrieved from the trust store settings on AEM Forms Server.

**statusMessage**

A `string` value that represents the revocation status message. The messages provide information about the reason for the revocation. For example, a message such as "Must sign the OCSP request" means that the OCSP response must be signed. The following are valid messages where *[Addition information provided.]* represents additional information provided by AEM Forms Server.

- OCSPNoCheck Extension is not allowed
- OCSP CertHash Extension is required
- OCSP CertHash in the response does not match the request certificate
- Must sign the OCSP request
- OCSP response signature is invalid
- OCSP request generation error: *[Addition information provided.]*
- OCSP request was null
- OCSP response parsing error: *[Addition information provided.]*
- OCSP transport error: *[Addition information provided.]*
- OCSP response has expired or is not yet valid
- OCSP response and request nonce does not match
- No CRL DPs found
- Unable to process a CRL DP: *[Addition information provided.]*
- Unable to retrieve CRL from: *[Addition information provided.]* with error:
- CRL thisUpdate is in the future
- CRL has expired or is not yet valid
- This is a delta CRL. Delta CRLs are not supported in this version.
- CRL parsing error: *[Addition information provided.]*
- CRL KeyID does not match
- CRL Authority Key ID extension is required
- CRL signature verification with issuer failed
- CRL Verification failure error: *[Addition information provided.]*
- CRL Issuer does not have a valid key usage
- No Valid CRL issuer found
- CRL or one of its entries contains an unrecognized critical extension
- No Valid CRL found in messages that can be returned:

**type**

A `string` value that represents the type of revocation information used. These string values are valid:

**CRL:**

Certificate Revocation List

**OCSP:**

Online Certificate Status Protocol

**validFrom**

A `dateTime` value that specifies the start date and time when the revocation is first valid.

**validTo**

A `dateTime` value that specifies the end date and time the revocation is valid. If this value is empty, the revocation information did not have a NextUpdate value present.

# 21.126. rights

A complex data type used to specify individual usage rights that are associated with a PDF document or credential. Rights variables are members of the `GetUsageRightsResult` data type.

For information about data that can be accessed using Xpath Expressions, see Dataitems.

## Data items

The data items that rights variables contain.

**enabledBarcodeDecoding:**

A `boolean` value that indicates whether the PDF document supports barcoded forms decoding within Adobe Reader.

**enabledComments:**

A `boolean` value that indicates whether the user can create and modify document annotations such as comments within Adobe Reader.

**enabledCommentsOnline:**

A `boolean` value that indicates whether the user can upload and download annotations such as comments to and from an online document review and comment server.

**enabledDigitalSignatures:**

A `boolean` value that indicates whether the user can digitally sign and save PDF documents and clear digital signatures within Adobe Reader.

**enabledDynamicFormFields:**

A `boolean` value that indicates whether the user can add, change, or delete fields and field properties on the PDF form within Adobe Reader.

**enabledDynamicFormPages:**

A `boolean` value that indicates whether the user can create pages from the form template for forms created in Acrobat.

**enabledEmbeddedFiles:**

A `boolean` value that indicates whether the user can add, remove, modify, or export files attached to a PDF document within Adobe Reader.

**enabledFormDataImportExport:**

A `boolean` value that indicates whether the user can add, remove, modify, or export files attached to a PDF document within Adobe Reader.

**enabledFormFillIn:**

A `boolean` value that indicates whether the user can fill form fields and save the PDF document locally from within Adobe Reader.

**enabledOnlineForms:**

A `boolean` value that indicates whether the user can access the database or call the web service that is defined within the PDF document.

**enabledSubmitStandalone:**

A `boolean` value that indicates whether the user can submit data to a server by email or offline.

# 21.127. RMInspectResult

A complex data type used to hold license information and other details retrieved from a protected document. `RMInspectResult` values are used to configure InspectProtected Document operations that the Rights Management service provides.

For information about data that can be accessed using Xpath Expressions, see Dataitems.

## Data items

The data items that `RMInspectResult` variables contain.

**alternateLicenseId**

A `string` value that represents the identifier of the alternate license used to protect the document

**docName**

A `string` value that represents the name of the policy-protected document.

**licenseId**

A `string` value that represents the identifier of the license used to protect the document.

**licenseIssuingAuthority**

A `string` value that represents the URL for the trusted security domain that issued the license. For example, `http://aps.adobe.com/edcws/services/urn:EDCLicenseService`.

**policyId**

A `string` value that represents the identifier of the policy used to protect the document.

**policyName**

A `string` value that represents the name of the policy used to protect the document.

**policySetId**

A `string` value that represents the identifier of the policy set which the policy belongs to.

**policySetName**

A `string` value that represents the name of the policy set which the policy belongs to.

**policyType**

A `string` value that represents the type of the policy used to protect the document.

**publisherId**

A `string` value that represents the identifier of the User Manager user who is the publisher of the document.

**publisherName**

A `string` value that represents the user name of the User Manager user who is the publisher of the document.

**publishTime**

A `date` value that represents the date the policy-protected document was published.

**revokeURL**

A `string` value that represents the URL to which a user is directed when they attempt to open a revoked document.

## 21.128. SearchQueryResultType

A complex data type that represents the result of a search operation. This data type is used in the `Search` operation provided by the Adobe Digital Enterpise Platform Connector for Microsoft SharePoint 10.

### Data Items

#### searchDocuments

A `list` value that represents the list of documents returned by the search operation.

#### searchCount

An `int` value that represents the number of results returned for the search.

#### totalAvailable

An `int` value that represents the total number documents available in the SharePoint repository.

#### searchStatus

A `string` value that represents the status of the search operation.

## 21.129. Rule

A complex data type that you can use as the value for the Pattern Matching Rules property of the operations that the Output service provides. It enables the creation of search rules that results in the Output service searching the input data and using a different form design based on the data content.

### Data items

The data items that `Rule` variables contain.

#### mPattern

A `string` value that represents the text pattern to search for.

#### mForm

A `string` value that represents the form design to use.

## 21.130. short

Holds a short int value (16-bit two's complement)

### Literal value

Any number between `-32768` and `32767`

### Example

```
12345
```

## 21.131. SignatureProperties

A complex data type that holds the information about a PDF signature. The `SignatureProperties` variable is used by the `PDFSignatureField` and `PDFSignatureVerificationResult` data types.

For information about data that can be accessed using Xpath Expressions, see Dataitems.

### Data items

The data items that `SignatureProperties` variables contain.

#### contactInfo

A `string` value that represents the contact information of the person who signed the PDF document.

#### legalAttestations

A `string` value that represents the legal attestations associated with signing the PDF document.

#### location

A `string` value that represents the location where the PDF document was signed.

#### reason

A `string` value that represents the reason that was provided for signing the PDF document.

#### revisionNumber

An `int` value that represents the revision number associated with the signature.

#### signerName

A `string` value that represents the name of the person who signed the PDF document.

#### signingDate

A `dateTime` value that represents the date and time that the PDF document was signed.

#### timestamp

A `Timestamp` value that represents the time the signature was signed.

**totalRevisions**

An `int` value that represents the total number of revisions associated with the signature.

# 21.132. SignatureType

A complex data type that stores information about the permissions and type of signature used to sign a PDF document. The SignatureType type is a member of the `PDFSignatureVerificationInfo` data type.

For information about data that can be accessed using Xpath Expressions, see Dataitems.

## Data items

The data items that `SignatureType` variables contain.

**permissions**

A `PDFSignatureVerificationResult` value that represents the actions that can be performed in the PDF document without invalidating the signature.

**type**

A `PDFSignatureType` value that specifies the type of signature that was applied to the PDF document.

# 21.133. Staple

A `string` value that represents the stapler options to use on the printer. `Staple` variables are used to configure the Staple property for the generatePrintedOutput operation of the Output service.

For information about data that can be accessed using Xpath Expressions, see Dataitems.

For information about configuring default properties, see Datatypespecific settings.

## Data items

`Staple` data values store one of these string values:

**off:**

Do not use the stapler on the printer.

**on:**

Use the stapler on the printer.

**usePrinterSetting:**

Use printer stapler settings.

### Datatype specific settings

Property for specifying whether to staple the printed output.

#### Default Value

Sets whether to staple the printed output. Select one of these values:

**off:**

Do not use the stapler on the printer.

**on:**

Use the stapler on the printer.

**usePrinterSetting:**

Use printer stapler settings.

## 21.134. string

Holds a string value.

For information about specifying the size of the variable, see Datatypedefinition.

For information about configuring default properties, see Datatypespecific settings.

### Datatype definition

Options for specifying the size of the variable.

**Maximum Length:**

Select this option to specify a maximum number of characters that the `string` variable can store.

**Unlimited:**

Select this option to specify that the `string` variable can store any amount of character data.

**IMPORTANT:** For installations of AEM Forms with an Oracle database, errors occur on AEM Forms Server when you change the maximum length of a `string`*variable to Unlimited for an activate process. The resulting errors deactivate the process and prevent it from becoming active again.*

If you need a `string` variable that has no size constraints and the process is already activated, re-create the `string` variable and select the Unlimited option in the variable properties. Do not change an existing `string` variable to have an unlimited size when a maximum length has been set.

### Datatype specific settings

Properties for specifying default property values for the variable.

**Default Value:**

The default length of the variable. Enter the text to use as the string value.

## 21.135. StyleGenerationLevel

A `string` value that represents the styles that are rendered with an HTML form. `StyleGenerationLevel` variables are used to configure the Style Generation Level property for the renderHTMLForm operation in the Forms service. The `StyleGeneration` data type is a data member of the HTMLRenderSpec data type.

For information about data that can be accessed using Xpath Expressions, see Dataitems.

For information about configuring default properties, see Datatypespecific settings.

### Data items

`StyleGeneration` data values store one of these string values:

**InlineAndInternalStyles:**

Use both inline and internal CSS styles. Internal styles use spacing and positional CSS styles. Inline styles use appearance styles, such as color, font, and background color.

**OnlyInlineStyles:**

Use only inline CSS styles from the custom CSS file. This option removes all internal styles inside the <head> tag on HTML pages.

**NoStyles:**

Do not use any CSS inline or internal styling. Only styles that are passed using an external CSS file are used.

### Datatype specific settings

Properties for setting the default values that the variable stores.

**Default value**

Select the default style generation level to store in the variable:

**InlineAndInternalStyles:**

Use both inline and internal CSS styles. Internal styles use spacing and positional CSS styles. Inline styles use appearance styles, such as color, font, and background color.

**OnlyInlineStyles:**

Use only inline CSS styles from the custom CSS file. This option removes all internal styles inside the <head> tag on HTML pages.

**NoStyles:**

Do not use any CSS inline or internal styling. Only styles that are passed using an external CSS file are used.

# 21.136. TaskContext

A complex data type that contains information about a task. The User service creates TaskContext values when it creates a task. TaskContext values are passed to Prepare Data services before the task asset is rendered. The values are useful for retrieving information about the task for use in the Prepare Data process. (See Aboutprepare data services.)

For information about data that can be accessed using XPath Expressions, see DataItems.

TaskContext values have no default properties to configure.

## Data Items

TaskContext values store the following values. The values represent information about the task with which the TaskContext value is associated.

### actionInstanceId

A `string` value that represents the unique identification of the run-time instance of the service operation that generated the task.

### assignedUser

A User value that represents the user who is assigned the task.

### assignedUserId

A `string` value that represents the unique identification of the user who is assigned the task.

### classOfTask

A `string` value that represents the type of task. The following values are valid:

**Multi:**

The Create Multiple Tasks operation generated the task.

**Standard:**

The Assign Task operation of the User 2.0 service created the task.

**Standard_LC8:**

The Assign Task operation of the User 1.0 service created the task. The task is the result of a legacy process that is running in the AEM Forms environment.

### createTime

A dateTime value that represents the date and time when the task was created.

### description

A `string` value that represents the text from the Description property of the Assign Task or Assign Multiple Tasks operation that generated the task.

### formURL

A `string` that represents the path of the application asset that is used with the task. The values of the Presentation And Data properties of the Assign Task or Assign Multiple Tasks operation that generated the task determine the value of formURL:

- When the Use An Application Asset option is selected, this value is the path that is specified for the Asset property.
- When the Use A Document Variable option is selected, this value is `null`.

### inputDocument

A document value that represents the document that is used as the presentation asset for the task. The values of the Presentation And Data properties of the Assign Task or Assign Multiple Tasks operation that generated the task determine the value of inputDocument:

- When the Use A Document Variable option is selected, this value is the value of the variable that is specified for the Variable property.
- When the Use An Application Asset option is selected, this value is `null`.

### isDraft

A boolean value that indicates whether the Workspace user who is assigned the task has saved a draft copy of the task. The value is `true` if the user has saved a draft copy. The value is `false` if the user has not saved a draft copy.

### isTaskActive

A boolean value that indicates whether the task is active or not. A value of `true` indicates the task is active. A value of `false` indicates the task is completed.

### processInstanceId

A long value that represents the unique identification of the process instance to which the task belongs.

### processName

A `string` value that represents the name of the process. This value is same as the value of the Name property of the process.

### routeList

A `string` value that represents a comma-separated list of the names of User Actions that are configured for the Assign Task or Assign Multiple Tasks operation. For example, if the operation has two User Actions named action1 and action2, the value of routeList is `action1,action2`.

### runtimeMap

A map of `string` values that represent run-time information that is provided to Workspace. The map contains the following useful keys:

**acrobatVersion:**

The version of Acrobat that that the user has installed.

**acroClientType:**

The client software that the user uses to open PDF documents, such as Acrobat or Adobe Reader.

**httpHeaders:**

The HTTP headers that are included in the message that is sent to Workspace on AEM Forms Server.

**relativeTargetUrl:**

The portion of the target URL after the IP address, for example `/workspace-server/submit`.

**targetUrl:**

The URL that the HTTP message is submitted to. For example the value of targetURL when submitting to AEM Forms Server on JBoss is `http://192.168.0.20:8080/workspace-server/submit`.

**userAgent:**

Information about the browser that is being used to open the task.

**TIP:** Use the runtimeMap data item to pass data from Prepare Data processes to custom Render processes. (See Aboutprepare data services.*)*

### selectedRoute

A `string` value that represents the name of the User Action that the user selects to complete the task. This value is `null` until the task is completed.

### serverReplyEmail

A `string` value that represents the email address that is configured for AEM Forms Server.

### stepName

A `string` value that represents the name of the operation on the process diagram that generated the task. This value is the same as the value of the Name property of the operation.

### taskId

A long value that represents the unique identification of the task.

### taskInstructions

A `string` value that represents the instructions that are provided with the task. This value is the same as the value of the Task Instructions property of the Assign Task or Assign Multiple Tasks operation that generated the task.

### taskStatus

A short value that represents the status of the task. This read-only value and can be one of the following numbers:

**1:**

The task has been created.

**2:**

The task has been created and saved. This status is assigned when a user opens a task from the Start Process page of Workspace and then saves a draft.

**3:**

The task is assigned to a user.

**4:**

The user has saved the task as a draft.

**100:**

The task has been completed.

**101:**

The task is completed because a deadline occurred.

**102:**

The task is completed because it was manually terminated using administration console.

### variationInputs

A map of string values that represent the property-value pairs to use for configuring a Guide variation. The map keys are the names of the variation input properties, and the associated values are the property

values. The values are configured in the Presentation and Data properties of the Assign Task or Assign Multiple Tasks operations. This value is `null` if no Guide variation is used.

**variationName**

A string value that represents the name of the Guide variation to use. This value is `null` if no variation is selected in the Presentation and Data properties of the Assign Task or Assign Multiple Tasks operations.

# 21.137. Task Date

A complex data type that defines a length of time. The User service uses Task Date values for calculating when task reminders, escalations, and deadlines occur:

- `Task Date` variables can be used as the value of the Schedule Escalation property of the AssignTask and AssignMultiple Tasks operations (User service).
- Task Reminder and Task Deadline data types include a Task Date value as a subtype.

`Task Date` values also indicate whether the business calendar is used when calculating dates.

You can create a Task Date variable, configure the default properties, and then use the variable as the value of the Schedule Escalation property of Assign Task or Assign Multiple Tasks operations. You can also use the Set Value service to add the value to a Task Reminder or Task Deadline value.

**TIP:** Create a Task Date variable only if you intend to use the value multiple times in the process. Otherwise, configure the Schedule Escalation property manually.

For example, a process includes a Task Date variable named taskDateVar and default values have been configured for the properties. You use XPath expressions with the Set Value service to add the value to a Task Reminder value named taskReminderVar:

```
/process_data/taskReminderVar/object/reminder =
/process_data/taskDateVar
```

Similarly, a Task Date value can be inserted into a Task Deadline value:

```
/process_data/taskDeadlineVar/object/dateObj = /process_data/taskDateVar
```

For information about data that can be accessed using Xpath Expressions, see Dataitems.

For information about configuring default properties, see Datatypespecific settings.

## Data items

`Task Date`values contain the following data items.

**Days**

A `string` value that represents the number of days.

**Hours**

A `string` value that represents the number of hours.

**Minutes**

A `string` value that represents the number of minutes.

**Total Minutes**

Do not use.

**UseBusinessDays**

A `boolean` value that determines whether a business calendar is used to perform date calculations.

### Datatype specific settings

You can configure default values for the following properties of `Task Date` variables.

**Use business calendar:**

Select to use the business calendar for date calculations.

**Days:**

Specify the number of days in the Task Date value.

**Hours:**

Specify the number of hours, in addition to the days, in the Task Date value.

**Minutes:**

Specify the number of minutes, in addition to the days and hours, in the Task Date value.

## 21.138. Task Deadline

A complex data type that defines a task deadline. These variables are used to configure task deadlines for AssignTask and AssignMultiple Tasks operations that the User service provides.

You can create a Task Deadline variable and use it as the value of the Deadline property of Assign Task and Assign Multiple Tasks operations. The operations use this property to calculate the date when the deadline occurs.

**TIP:** Create a Task Deadline variable only if you intend to use the value multiple times in the process. Otherwise, configure the Deadline property manually.

For information about data that can be accessed using Xpath Expressions, see Dataitems.

For information about configuring default properties, see Datatypespecific settings.

## Data items

`Task Deadline` values contain the following data items.

### dateObj

A `TaskDate` value that represents the amount of time that passes after the task is assigned before the deadline occurs.

### changeInstructions

A `boolean` value that indicates whether the task instructions should be changed when the deadline occurs. A value of `true` indicates the instructions are changed, and `false` indicates they are not changed.

### dateObj

A `TaskDate` value that determines the amount of time after the task is created when the deadline occurs.

### deadlineInstructions

A `string` value that contains the new task instructions to apply when the deadline occurs. This value is used only if the value of the changeInstructions property is `true`.

### followRouteOnDeadline

A `boolean` value that can enable following a specific route from the Assign Task operation when the deadline occurs.

### selected

A `boolean` value that determines whether a deadline is enabled or disabled.

### selectedRoute

A `string` value that represents the name of the route to follow after the Assign Task or Assign Multiple Tasks operation. AEM Forms Server sets this value when the user submits their task.

## Datatype specific settings

Properties for configuring the default values of `Task Deadline` properties. You configure these properties in the same way that you configure the Deadline property of Assign Task operations. (See Deadline.)

# 21.139. Task Delegate and Consult

A complex data type that holds delegate and consult information for tasks. These variables are used to configure AssignTask and AssignMultiple Tasks operations that the User service provides.

You can create a Task Delegate and Consult variable and use it as the value of the Forward and Consult property of Assign Task and Assign Multiple Tasks operations. This property determines whether tasks can be forwarded or consulted with other users. It also specifies the user group that can be forwarded to or consulted with. For more information see Configuringtask delegations and consultations.

**TIP:** Create a Task Delegate and Consult variable only if you intend to use the value multiple times in the process. Otherwise, configure the Forward and Consult property manually.

For information about data that can be accessed using Xpath Expressions, see Dataitems.

For information about configuring default properties, see Datatypespecific settings.

## Data items

`Task Delegate and Consult` values contain the following data items.

### canConsultOnlytoGroup

A `boolean` value that indicates whether the consult option should be restricted to a specific group of users. If `true`, the task can only be consulted with a member of a specified group.

### canConsultTask

Do not use. This value is used by AEM Forms Server.

### canForwardOnlytoGroup

A `boolean` value that indicates whether the forward option should be restricted to a specific group of users. If `true`, the task can only be forwarded to a member of a specified group.

### canForwardTask

Do not use. This value is used by AEM Forms Server.

### consultNfo

A `TaskUser Selection` value that identifies the users that can be consulted with for the task.

### forwardNfo

A `TaskUser Selection` value that identifies the users that the task can be forwarded to.

### Datatype specific settings

Use these properties to configure the default value of `Task Delegate and Consult`variables. You configure these properties in the same way that you configure the Forward And Consult property of Assign Task and Assign Multiple Tasks operations. (See ReassignmentRestrictions.)

## 21.140. Task Priority

A complex data type that represents the priority of a task. These variables are used to configure the Task Priority property of AssignTask and AssignMultiple Tasks operations that the User service provides.

You can create a Task Priority variable and use it as the value of the Task Priority property of Assign Task and Assign Multiple Tasks operations. The operations use this property to prioritize the task in Workspace To Do lists. (See Specifyingtask priority.)

**TIP:** Create a Task Priority variable only if you intend to use the value multiple times in the process. Otherwise, configure the Task Priority property manually.

For information about data that can be accessed using Xpath Expressions, see Dataitems.

For information about configuring default properties, see Datatypespecific settings.

### Data items

`Task Priority` values contain the following data item.

#### prioritySelection

A `short` value that indicates the priority for the task. These values are valid:

**1:**

Highest priority

**2:**

High priority

**3:**

Normal priority

**4:**

Low priority

**5:**

Lowest priority

### Datatype specific settings

Properties for configuring variable default values. You configure these properties in the same way that you configure the Priority properties of Assign Task operations. (See TaskRoutes and Priority.)

## 21.141. Task Reminder

A complex data type that defines a task reminder. Task Reminder variables are used to configure task reminders for AssignTask and AssignMultiple Tasks operations that the User service provides.

You can create a Task Reminder variable and use it as the value of the Reminder property of Assign Tasks and Assign Multiple Tasks operations. The operations use this property to calculate the date when the first and repeating reminders occur. (See Sendingreminders about tasks.)

**TIP:** Create a Task Reminder variable only if you intend to use the value multiple times in the process. Otherwise, configure the Reminder property manually.

For information about data that can be accessed using Xpath Expressions, see Dataitems.

For information about configuring default properties, see Datatypespecific settings.

### Data items

`Task Reminder` values contain the following data items.

#### changeInstructions

A `boolean` value that indicates whether the task instructions are changed when the reminder occurs. A value of `true` indicates the instructions are changed, and `false` indicates they are not changed.

#### Reminder

A `TaskDate` value that represents the amount of time that passes after the task is assigned before the initial reminder occurs.

#### reminderInstructions

A `string` value that specifies the new task instructions to apply when the reminder occurs. This value is used only if the value of the `changeInstructions` data item is `true`.

#### repeatSelected

A `boolean` value that determines whether reminders are repeated. A value of `true` indicates reminders are repeated, and `false` indicates the reminder occurs once.

#### repeatReminder

A `TaskDate` value that determines the amount of time that passes between repeated reminders.

### selected

A `boolean` value that determines if reminders are enabled.

## Datatype specific settings

Use these properties for configuring the default values of `Task Reminder` variables. You configure these properties in the same way that you configure the Reminder properties of Assign Task and Assign Multiple Tasks operations. (See Reminders.)

# 21.142. Task Result

A complex data type that represents the information that is submitted to AEM Forms Server when a forms workflow task is submitted. Task Result values are saved by using the Task Result property of AssignTask operations.

For information about data that can be accessed using Xpath Expressions, see Dataitems.

## Data items

The data items that Task Result values contain.

### actionInstanceId

A `long` value that represents the unique identification of the operation instance that produced the Task Result value.

### assignedDate

A `dateTime` value that represents the date and time when the task was assigned to the user.

### attachmentList

A `list` of `document` values that represent the attachments and notes that the user added to the task.

### completedBy

A `com.adobe.idp.um.api.infomodel.Impl.PrincipalImpl` value that represents the user who completed the task.

### completedById

A `string` value that represents the unique identification of the user who completed the task.

### completedDate

A `dateTime` value that represents the date and time when the user completed the task.

**completionNotes**

A `string` value that represents the comments that the user added to the task. Comments are added using the Approval Container (Deprecated) tools.

**formDataDocument**

A `document` value that contains the field data that was submitted with the task.

**formDataXML**

An `xml` value that represents the field data that was submitted with the task.

**initialPrincipal**

A `com.adobe.idp.um.api.infomodel.Impl.PrincipalImpl` value that represents the user who was initially assigned the task.

**initialPrincipalId**

A `string` value that represents the unique identification of the user who was initially assigned the task.

**ipAddress**

A `string` value that represents the IP address of the computer that was used to complete the task.

**isActive**

A `boolean` value that indicates whether the task is active or completed. A value of true indicates the task is active and false indicates the task is completed.

**isCompleted**

A `boolean` value that indicates whether the task is active or completed. A value of true indicates the task is completed and false indicates the task is active.

**isDeadlined**

A `boolean` value that indicates whether a deadline caused the task to complete. A value of true indicates a deadline occurred and false indicates no deadline occurred.

**isTerminated**

A `boolean` value that indicates whether the task was completed by being terminated. A value of true indicates the task was terminated and false indicates the task was not terminated.

**possibleUserActions**

A `list` of `string` values that represent the user actions that were available for completing the task.

### selectedUserAction

A `string` value that represents the user action that was used to complete the task.

### stepName

A `string` value that represents the name of the operation that generated the task. This value is the Name property of the Assign Task or Assign Multiple Tasks operation on the process diagram.

### taskId

A `long` value that represents the unique identification of the task that produced the Task Result value.

### taskStatus

A `short` value that represents the status of the task. The following values are possible:

**100:**

The task was completed normally.

**101:**

A deadline caused the task to complete.

**102:**

The task was terminated.

RELATED LINKS:

Saving task data

Task Result Collection

## 21.143. Task Result Collection

A complex data type that stores the results of one or more tasks. This type also provides a list of user actions that were selected to complete the tasks.

- Assign Multiple Tasks operations store the results of the tasks that it generates in Task Result Collection values. Task Result Collection variables are used to configure the Output properties of AssignMultiple Tasks operations.
- The results of Assign Task operations can also be stored in Task Result Collection values. Task Result Collection variables are used to configure the Task Result Collection property of AssignTask operations.

For information about data that can be accessed using Xpath Expressions, see Dataitems.

## Data items

### size

An `int` value that represents the number of Task Result values that are stored in the Task Result Collection value.

### taskResults

A `list` of `TaskResult` values.

### userActions

A `list` of `string` values that represent the names of the user actions that were presented to users for completing the tasks.

RELATED LINKS:

Saving task data

Providing actions for submitting tasks

# 21.144. Task Routes and Priority

**NOTE:** This variable type is deprecated in AEM Forms 9.0.

A complex data type that holds information about routes and task priority used with AssignTask operation operations that the User service provides.

For information about data that can be accessed using Xpath Expressions, see Dataitems.

For information about configuring default properties, see Datatypespecific settings.

## Data items

The data items that `Task Routes and Priority` variables contain.

### initTaskWithRoute

A `boolean` value that indicates whether to initialize the routes from the Assign Task operation to the task.

### mustSelectRouteName

A `boolean` value that indicates whether the end user is required to chose a route when completing the task.

### prioritySelection

A `short` value that indicates the priority for the task. These values are valid:

**1:**

Highest priority

**2:**

High priority

**3:**

Normal priority

**4:**

Low priority

**5:**

Lowest priority

### Datatype specific settings

Properties for configuring variable default values. You configure these properties in the same way that you configure the Routes And Priority properties of Assign Task operations. (See TaskRoutes and Priority and Specifyingtask priority.)

## 21.145. Task Runtime UI

A complex data type that represents the configuration of the Workspace User Interface properties of AssignTask and AssignMultiple Tasks operations (User 2.0 service).

For information about data that can be accessed using Xpath Expressions, see Dataitems.

For information about configuring default properties, see Datatypespecific settings.

### Data items

Data items that Task Runtime UI values contain.

#### approvalContainerUI

A `boolean` value that indicates whether the Workspace Approval Container (Deprecated) is used. A value of `true` indicates the Workspace Approval Container (Deprecated) is used, and `false` indicates it is not used.

If this value is `true`, the value for the customUI data item must be `false`.

#### customUI

A `boolean` value that indicates whether a custom UI is used. A value of `true` indicates custom UI is used, and `false` indicates it is not used.

If this value is `true` and the value for approvalContainerUI is `true`, customUI is used.

**mustOpenFormToComplete**

A `boolean` value that indicates whether the task must be opened to complete it. A value of `true` indicates the task must be opened, and `false` indicates the task can be completed from the task list.

**openFormFullScreen**

A `boolean` value that indicates whether the task opens in full-screen mode in Workspace. A value of `true` indicates the task opens in full screen mode, and `false` indicates it opens normally.

**tloPath**

A `string` value that represents the path to the SWF file that implements the customuser interface.

**type**

An enumeration that indicates the Workspace user interface to display. The following `string` values are valid:

**DEFAULT:**

Display the default user interface.

**APPROVAL:**

Display the Approval Container (Deprecated) user interface.

**CUSTOM:**

Display a custom user interface.

### Datatype specific settings

Properties for configuring variable default values. You configure these properties in the same way that you configure the Workspace User Interface properties of Assign Task operations. (See WorkspaceUser Interface.)

RELATED LINKS:

Make opening tasks optional

Specifying the Workspace user interface

Maximizing the form or Guide

## 21.146. Task User Selection

A complex data type that represents a user group that a task can be forwarded to or consulted with. Task Delegate and Result data types include Task User Selections as a subtype. (See TaskDelegate and Consult.)

Create a Task User Selection variable, configure the default properties, and use the Set Value service to add the value to Task Delegate and Consult values. Task Delegate and Consult values include the entire data model of Task User Selection data types. However, Task Delegate and Consult values use only the data items that relate to user groups.

**TIP:** Create a Task User Selection variable only if you intend to use the value multiple times in the process. Otherwise, configure the Task Delegate and Consult variable manually.

For example, a process includes a Task User Selection variable named taskUSVar and default values have been configured for the properties. You use XPath expressions with the Set Value service to add the value to a Task Delegate and Consult value named taskDGVar. The same user group is used to consult and forward tasks:

```
/process_data/taskDGVar/object/forwardNfo = /process_data/taskUSVar
/process_data/taskDGVar/object/consultNfo = /process_data/taskUSVar
```

For information about configuring the default values of `Task Data` variables, see Datatypespecific settings. For information about the data items that Task Date values include, see Dataitems.

For information about data that can be accessed using Xpath Expressions, see Dataitems.

For information about configuring default properties, see Datatypespecific settings.

### Data items

`Task User Selection` values contain the following data items.

#### domainId

A `string` value that specifies the domain name that the group resides in.

#### canonicalName

A `string` value that specifies the canonical name for the group.

#### All other data items

Not used with Task Delegate and Consult values.

### Datatype specific settings

Use these properties for configuring the default values of `Task User Selection` variables. You configure these properties in the same way that you configure the Initial User Selection properties of Assign Task operations. (See InitialUser Selection.)

## 21.147. TemplateSearchFilter

A complex data type that represents review template attributes to search for. Used with the Review, Commenting, and Approval building block, which is installed with the Managed Review & Approval Solution Accelerator.

When more than one criterion is used, an AND condition joins them. Use % (per cent) character for wild card searches. For example, templateName is `Test%` and complianceCode is `SOP`. This search returns all the templates in which both the name starts with Test and complianceCode is SOP.

You can also paginate search results using pageSize and pageNumber. Use pageSize to define the maximum number of rows that are returned in the result set. Use pageNumber to retrieve subsequent of the search result. By default, pageNumber is 0, which means all the rows in the result are returned. To retrieve the last page of the result set for a given pageSize, set pageNumber as (@see LAST_PAGE}. If paging is not required, do not set these attributes, or set one or both to 0.

For information about data that can be accessed using XPath Expressions, see Dataitems.

## Data items

The data items that TemplateSearchFilter values contain.

### author

A `string` value that represents the author of the review template.

### complianceCode

A `string` value that represents the legislation that this review complies with.

### customAttributes

A `map` of CustomAttributeTO values that represent custom template attributes to search for.

### keywords

A `string` value that represents words to search for in the template description.

### pageNumber

An `int` value that represents a page number of the search results.

### pageSize

An `int` value that represents the number of rows per page to use when calculating pagination.

### retrieveActiveOnly

A `boolean` value that indicates whether to search active templates only. If `true`, search returns only active templates.

### templateName

A `string` value that represents the template name to search for.

# 21.148. Timestamp

A complex type that represents information about the timestamp and identity of the person who signed the PDF document. `Timestamp` variables are members of the `SignatureProperties` data type.

For information about data that can be accessed using Xpath Expressions, see Dataitems.

## Data items

The data items that `Timestamp` variables contain.

### timestamp

A `byte` value that specifies the time that the timestamp occurred.

### timestamped

A `boolean` value that specifies that the PDF signature has a timestamp.

### timestampInformation

An `IdentityInformation` value that specifies the identity of the person who signed the PDF document.

# 21.149. Timezone

A complex data type used to specify a time zone. `Timezone` variables are members of the `Owner` data type.

For information about data that can be accessed using Xpath Expressions, see Dataitems.

## Data items

The data items that`timezone`variables contain.

### displayName

A `string` value that represents the name of the time zone that appears to the user.

### DSTSavings

A `int` value that represents the amount of time to be added to local standard time to get local wall clock time.

### ID

A `string` value that represents the identifier of the time zone.

### rawOffset

A `int` value that represents the amount of time in milliseconds to add to UTC (Coordinated Universal Time) to get standard time for this time zone.

# 21.150. ToImageOptionsSpec

A complex data type that you can use as the value for the Convert Image Options property of the toImage(deprecated) and toImage operations that the Convert PDF service provides.

The information that this variable holds specifies properties of the image files that the toImage operation generates, such as the type of image compression, color format, and resolution.

For information about data that can be accessed using Xpath Expressions, see Dataitems.

## Data items

The data items that `ToImageOptionsSpec` variables contain.

### cmykPolicy

A `string` value that represents the CMYK policy. This option is valid for conversions to JPEG, JP2K, and TIFF image formats. These values are valid:

- embedProfile
- off

### colorCompression

A `string` value that represents the color compression to use. This option is valid for conversion to JPEG, JP2K, and TIFF formats. These values are valid:

- minimum
- low
- medium
- high
- maximum
- lossless
- lzw
- zip
- none

### colorSpace

A `string` value that represents the color space. These values are valid:

- rgb
- cmyk

- grayscale

**filter**

A `string` value that represents the filter. This option is valid for conversion to PNG format only. These values are valid:

- none
- sub
- up
- average
- path
- adaptive

**format**

A `string` value that represents the JPEG format. This option is valid for conversion to JPEG and JP2K formats. These values are valid:

- standard
- optimized
- 3scans
- 4scans
- 5scans

**grayScaleCompression**

A `string` value that represents the grayscale compression. This option is valid for conversion to JPEG, JP2K, and TIFF formats. These values are valid:

- minimum
- low
- medium
- high
- maximum
- lossless
- lzw
- zip
- none

**grayScalePolicy**

A `string` value that represents the grayscale policy. This option is valid for conversion to JPEG, JP2K, and TIFF formats. These values are valid:

- embedProfile
- off

### imageConvertFormat

A `string` value that represents the image conversion format. This property must be set. These values are valid:

- jpeg
- jp2k
- png
- tiff

### interlace

A `string` value that represents the interlace. This option is valid for conversion to PNG format only. These values are valid:

- none
- adam7

### monochrome

A `string` value that represents the monochrome compression. This option is valid for conversion to TIFF format only. These values are valid:

- none
- ccittg3
- ccittg4
- lzw
- zip

### options

A `string` value that represents the options that are set.

### resolution

A `string` value that represents the image resolution. These values are valid:

- 72
- 96
- 150
- 300
- 600
- 1200
- 2400

### rgbPolicy

A `string` value that represents the RGB policy. These values are valid:

- embedProfile

- off

**rowsPerStrip**

An `int` value that represents the number of rows of pixels per strip to use for encoding the TIFF image. A value of `0` sets the rows per strip equal to the image length, resulting in a single strip. A value of `-1` (the default value) sets the rows per strip equal to infinity, resulting in a single strip.

This value is only used when the imageConvertFormat property is set to `tiff`.

**tileSize**

An `int` value that represents the image tile size. This option is valid for conversion to JP2K format only. The value must be within the range `128-2048`, inclusive.

# 21.151. ToPSOptionsSpec

A complex data type that represents various preferences used to convert the PDF document to a PostScript file. This variable type is used by the toPS2 operation in the Convert PDF service.

For information about data that can be accessed using Xpath Expressions, see Dataitems.

## Data items

The data items that `ToPSOptionsSpec` variables contain.

**allowBinaryContent**

A `boolean` value that indicates whether to create the PostScript file in binary format. A value of `true` means to create the file in binary format. The default value of `false` means not to create the file in binary format.

**bleedMarks**

A `boolean` value that indicates whether to place a mark in each of the four corners to indicate the PDF bleed box boundaries. A value of `true` means to place the bleed marks. The default value of `false` means not to place the bleed marks.

**color**

A `string` value that indicates whether to use color in the PostScript file.

These values are valid:

**Composite:**

Produce a PostScript file that includes color.

**Composite Gray:**

Produce a PostScript file that does not include color.

The default value is `Composite`.

**colorBars**

A `boolean` value that indicates whether to place a color bar at the top of the page, outside the crop area. A color bar shows a single box for each spot or process color in the document. A value of `true` means to place a color bar. The default value of `false` means not to place a color bar.

**convertTrueTypeToType1**

A `boolean` value that indicates whether to convert TrueType fonts to Type 1 fonts in the resulting PostScript file. A value of `true` means to perform the conversion. The default value of `false` means not to perform the conversion.

**emitCIDFontType2**

A `boolean` value that indicates whether to preserve the hinting information in the original font in the resulting PostScript file. A value of `true` means to preserve the hinting information in the original font. The default value of `false` means that the CIDFontType2 fonts are converted to CIDFontType0 fonts, resulting in compatibility with a wider range of printers.

**emitPSFormObjects**

A `boolean` value that indicates whether to emit PostScript form objects for Form XObjects found in the PDF document. The Form XObjects are used to represent complex objects, such as background images, that appear multiple times in the PDF file. The value of `true` means that the PostScript form objects are created. Although this setting reduces the size of the print job, it increases the printer memory that is used. The default value of `false` means that the PostScript form objects are not created.

**expandToFit**

A `boolean` value that indicates whether to expand the contents of the input PDF file to fit the selected page size. A value of `true` indicates to expand the contents. The default value of `false` indicates not to expand the contents.

**fontInclusion**

A `string` value that represents the fonts to be embedded in the resulting PostScript file. These values are valid:

**None:**

The system fonts are used instead of the embedded fonts and referenced fonts.

**Embedded Fonts:**

The fonts used are taken directly from the PDF file.

**Embedded Fonts and Reference Fonts:**

The fonts used are taken from the PDF file and the system in use.

The default value is `Embedded Fonts`.

**includeComments**

A `boolean` value that indicates whether to preserve the appearance of comments in the resulting PostScript file. A value of `true` means to preserve the appearance of comments in the resulting file. The default value of `false` means not to preserve the appearance of comments in the resulting file.

**lineWeight**

An `int` value that specifies the weight of the lines used for the trim marks, bleed marks, and registration marks. The valid values are 0.125 pt, 0.25 pt, and 0.50 pt. The default value is 0.25 pt.

**pageInformation**

A `boolean` value that indicates whether to place page information outside the crop area of the page. A value of`true` means to place page information outside the crop area. The default value of `false` means not to place page information outside the crop area.

**pageRange**

An `int` value that represents the page numbers in the PDF document to be converted to a PostScript file. Single page numbers separated by commas or a range of pages, or both, can be specified (for example, `22, 7-9, 13`).

**pageSize**

A `string` value that represents the page size of the PostScript file output. The default value is DetermineAutomatically. These values are valid:

**A2, A3, A4, A5, Envelop, Executive, Folio, Legal, Letter, Tabloid**

**DetermineAutomatically:**

The service determines the page size to use.

**Custom:**

When specified, the pageSizeHeight and pageSizeWidth items should be set.

**pageSizeHeight**

A `string` value that represents the height of the page in the PostScript file. The height is specified using the format *[integer][measurement unit]*, where *[measurement unit]* can be `pt`, `in`, `mm`, or `cm` (for example, `10mm`). The default measurement unit is `pt`.

This value should be set when pageSize is specified as Custom.

### pageSizeWidth

A `string` value that represents the width of the page in the PostScript file. The width is specified using the *[integer][measurement unit]* format, where *[measurement unit]* can be `pt`, `in`, `mm`, or `cm` (for example, `10mm`). The default measurement unit is `pt`.

This value should be set when pageSize is specified as `Custom`.

### psLevel

A `string` value that represents the PostScript language compatibility. The default value is `Language Level 2`. These values are valid:

**Language Level 2:**

The PostScript language that was released in 1992. It supports color printing.

**Language Level 3:**

The PostScript language that was released in 1997. It supports more fonts, has better graphics handling, and includes several features to speed up PostScript printing.

### registrationMarks

A `boolean` value that indicates whether to place a registration mark outside the crop area of the page. A value of `true` means to place a registration mark. The default value of `false` means not to place a registration mark.

### resolution

An `int` value that represents the resolution of the output in dots per inch (dpi). The default value is `300`.

### reverse

A `boolean` value that indicates whether to reverse the contents of the input PDF file in the page. A value of `true` means to reverse the contents. The default value of `false` means not to reverse the contents.

### rotateAndCenter

A `boolean` value that indicates whether to rotate and center the contents of the input PDF file in the selected page. A value of `true` means to center and rotate the contents. The default value of `false` means not to center or rotate the contents.

### shrinkToFit

A `boolean` value that indicates whether to shrink the contents of the input PDF file to fit the selected page size. A value of `true` means to shrink the contents. The default value of `false` means not to shrink the contents.

### style

A `string` value that represents the style of the printer marks to create. The default value is `Default`.

These values are valid:

- Default
- InDesignJ1
- InDesignJ2
- Illustrator
- IllustratorJ
- QuarkXPress

**trimMarks**

A `boolean` value that indicates whether to place a mark in each of the four corners to indicate the PDF trim box boundaries. A value of `true` means to include the trim marks. The default value of `false` means not to include the trim marks.

**useMaxJPEGImageResolution**

A `boolean` value that indicates whether to use the highest resolution for printing JPEG 2000 images. The default value of `true` means to use the highest resolution possible. A value of `false` means not to use the highest resolution possible.

# 21.152. ToSWFOptionsSpec

A complex data type that you can use as the value for the Convert SWF Options property of the toSWF operation that the Convert PDF service provides.

The information that this variable holds specifies properties of the SWF files that the toSWF operation generates.

## Data items

The data items that ToSWFOptionsSpec variables contain.

**setHeight**

A double value that represents the height of the SWF file. A valid value is >=0.

**setWidth**

A double value that represents the width of the SWF file. A valid value is >=0.

**setVersion**

An enum value that represents the version with which the SWF file will be encoded. A valid value is v_1 through v_10.

**setSampleX**

A double value that represents the down-sampling factor of the image in x direction. A valid value is 0.0 to 1.0.

**setSampleY**

A double value that represents the down-sampling factor of the image in y direction. A valid value is 0.0 to 1.0.

**setProcessSignatures**

A boolean value that represents whether signed signature fields are processed during conversion. A valid value is true or false.

**setPageRange**

A string value that represents the pages to be converted to SWF. A valid value is page numbers or range separated by a comma (,). For example: 1, 5-7 means pages 1, 5, 6, and 7.

# 21.153. TransferModeEnum

A `string` value that represents the format of the file data that is transferred when using the Get, Getmultiple documents, Getlist of files, Getto file system, Put, Putfrom file system, and Putmultiple documents that the FTP service provides. The valid values are `Binary` and `ASCII`.

ASCII mode transfers files faster than Binary mode.

# 21.154. TransformationFormat

A `string` value that represents the type of PDF to generate. `TransformationFormat` variables are used to configure the Transformation Format property for the generatePDFOutput(deprecated), generatePDFOutput, and transformPDF of the Output service.

For information about data that can be accessed using Xpath Expressions, see Dataitems.

For information about configuring default properties, see Datatypespecific settings.

## Data items

`TranformationFormat` data values store one of these string values:

**PDF:**

A non-interactive PDF document is created.

**PDFA:**

A non-interactive PDF/A document is created. The PDF/A value specifies a W3C Archive Standard PDF/A document that can be used for archiving purposes. It also embeds all the fonts as well as turns off compression.

### Datatype specific settings

Property for specifying the format for generating a PDF document:

**Default value**

Sets the default format for generating the PDF document. Select one of these values:

**PDF:**

A non-interactive PDF document is created.

**PDFA:**

A non-interactive PDF/A document is created. The PDF/A value specifies standard for archiving purposes which embeds the fonts and turns off compression.

## 21.155. TransformTo

A `string` value that represents the HTML transformation types. `TransformTo` variables are used to configure the Transform To property in the renderHTMLForm(deprecated) and renderHTMLForm operations of the Forms service.

For information about data that can be accessed using Xpath Expressions, see Dataitems.

For information about configuring default properties, see Datatypespecific settings.

### Data items

`TransformTo` data values store one of these string values:

**AHTML:**

Generates HTML that is compatible with accessibility-enhanced browsers (Internet Explorer 5 or later).

**AUTO:**

The Forms service determines the optimal transformation to perform.

**HTML4:**

Generates HTML that is compatible with older browsers that do not support absolute positioning of HTML elements.

**MSDHTML:**

Generates HTML that is compatible with dynamic HTML for Internet Explorer 5.0 or later.

**NoScriptXHTML:**

Generates HTML that is compatible with the CSS2 specification and compliant with XHTML 1.0.

**StaticHTML:**

Generates HTML that does not allow users to enter data in the fields of HTML forms.

**XHTML:**

Generates HTML that is compatible with accessibility-enhanced browsers (Internet Explorer 5 or later).

**AccessibleXHTML:**

Generates HTML that is XHTML 1.0 standard-complaint. Elements in the HTML form are positioned using the relative em units. Using relative units helps to ensure that an HTML form appears correctly when tools are used to make content accessible to users with visual impairment.

**NoScriptAccessibleXHTML:**

Generates HTML that is standard-compliant with the XHTML 1.0 and CSS2 specifications. The XHTML also does not contain any client-side JavaScript, although server-side JavaScript can be included. Elements in the HTML form are positioned using the `em` tag, which is a relative unit. Using relative units helps to ensure that an HTML form appears correctly when tools are used to make content accessible to users with visual impairment.

## Datatype specific settings

Properties for setting the default values that the variable stores.

### Default value

Select the default transformation type to store in the variable:

**AUTO**

The Forms service determines the optimal transformation to perform.

**XHTML:**

Generates HTML that is compatible with accessibility-enhanced browsers (Internet Explorer 5 or later).

**NoScriptXHTML:**

Generates HTML that is compatible with the CSS2 specification and compliant with XHTML 1.0.

**HTML4:**

Generates HTML that is compatible with older browsers that do not support absolute positioning of HTML elements.

**MSDHTML:**

Generates HTML that is compatible with dynamic HTML for Internet Explorer 5.0 or later.

**AHTML:**

Generates HTML that is compatible with accessibility-enhanced browsers (Internet Explorer 5 or later).

**StaticHTML:**

Generates HTML that does not allow users to enter data in the fields of HTML forms.

**AccessibleXHTML:**

Generates HTML that is XHTML 1.0 standard-complaint. Elements in the HTML form are positioned using the relative `em` units. Using relative units helps to ensure that an HTML form appears correctly when tools are used to make content accessible to users with visual impairment.

**NoScriptAccessibleXHTML:**

Generates HTML that is standard-compliant with the XHTML 1.0 and CSS2 specification. The XHTML also does not contain any client-side JavaScript, though server-side JavaScript can be included. Elements in the HTML form are positioned using the `em` tag, which is a relative unit. Using relative units helps to ensure that an HTML form appears correctly when tools are used to make content accessible to users with visual impairment.

## 21.156. TransportSecurityEnum

A `string` value that represents the transport security used by the Email service when communicating with the POP3 or IMAP server. The value depends on how the email server is configured.

These values are valid:

**None:**

Clear text

**SSL:**

Secure Sockets Layer

**TSL:**

Transport Layer Security

For information about configuring default properties, see Datatypespecific settings.

### Datatype specific settings

A list of valid transport securities used for communicating with the POP3 or IMAP servers.

## 21.157. TSPOptionSpec

A complex data type that stores preferences for the time-stamping provider (TSP) support.

`TSPOptionSpec` variables are used in the following operations of the Signature service:

CertifyPDF

SignSignature Field

VerifyPDF Signature

VerifyPDF Signature (deprecated)

For information about data that can be accessed using Xpath Expressions, see Dataitems.

For information about configuring default properties, see Datatypespecific settings.

### Data items

The data items that `TSPOptionSpec` variables contain.

#### tspHashAlgorithm

A `string` value that represents a hash algorithm associated with the TSP.

These string values are valid:

**SHA1:**

The Secure Hash Algorithm that has a 160-bit hash value.

**SHA256:**

The Secure Hash Algorithm that has a 256-bit hash value.

**SHA384:**

The Secure Hash Algorithm that has a 384-bit hash value.

**SHA512:**

The Secure Hash Algorithm that has a 512-bit hash value.

**RIPEMD160:**

The RACE Integrity Primitives Evaluation Message Digest that has a 160-bit message digest algorithm and is not FIPS-compliant.

The default value is `SHA1`.

### tspRevocationCheckStyle

A `string` value that represents the type of revocation checks performed when verifying a signature in a PDF document.

These string values are valid:

**AlwaysCheck:**

Checks for revocation of all certificates.

**BestEffort:**

Checks for revocation of all certificates when possible.

**CheckIfAvailable:**

Checks for revocation of all certificates only when revocation information is available.

**NoCheck:**

Does not check for revocation.

The default value is `BestEffort`.

### SendNonce

A `boolean` value that indicates whether a nonce is sent with this TSP request. A *nonce* can be a time-stamp, a visit counter on a web page, or a special marker. The parameter is intended to limit or prevent the unauthorized replay or reproduction of a file. A value of `true` indicates that a nonce is sent with the TSP request and `false` indicates that nonce is not sent.

### tspServerPassword

A `string` value that represents a password for accessing the TSP server using the specified user name.

### tspServerURL

A `string` value that represents the URL for the TSP server. If no value is provided, the timestamp from the local system is applied.

### tspServerUserName

A `string` value that represents a user name for accessing the TSP server.

### tspSize

An `int` value that represents the estimated size of the TSP request in bytes. Valid values are from `60` to `10240`. The default value is `4096`.

### useExpiredTimestamps

A `boolean` value that indicates whether to use a timestamp that has expired. A value of `False` means to not use expired timestamps. The default is `True`, which means to use expired timestamps during validation of the certificate.

## Datatype specific settings

Properties for configuring time-stamping information applied to the certified signature.

### Time Stamp Server URL

Sets the URL for a TSP server. If no value is provided, the timestamp from the local system is applied.

### Time Stamp Server Username

Sets the user name if necessary for accessing the TSP server.

### Time Stamp Server Password

Sets the password for the user name if necessary for accessing the TSP server.

### Time Stamp Server Hash Algorithm

Sets the hash algorithm used to digest the request sent to the timestamp provider. The default value is SHA1. Select one of these values:

**SHA1:**

The Secure Hash Algorithm that has a 160-bit hash value.

**SHA256:**

The Secure Hash Algorithm that has a 256-bit hash value.

**SHA384:**

The Secure Hash Algorithm that has a 384-bit hash value.

**SHA512:**

The Secure Hash Algorithm that has a 512-bit hash value.

**RIPEMD160:**

The RACE Integrity Primitives Evaluation Message Digest that has a 160-bit message digest algorithm and is not FIPS-compliant.

### Revocation Check Style

Sets the revocation-checking style used for verifying the trust status of the CRL provider's certificate from its observed revocation status. The default value is BestEffort. Select one of these values:

**NoCheck:**

Does not check for revocation.

**BestEffort:**

Checks for revocation of all certificates when possible.

**CheckIfAvailable:**

Checks for revocation of all certificates only when revocation information is available.

**AlwaysCheck:**

Checks for revocation of all certificates.

**Use Expired Timestamps**

Select this option to use timestamps that have expired during the validation of the certificate. When this option is deselected, expired timestamps are not used. By default, this option is selected.

**Predicted Time Stamp Token Size (In Bytes)**

Sets the estimated size, in bytes, of the TSP response. The size is used to create a signature hole in the PDF document. This value represents the maximum size of the timestamp response that the configured TSP could return. Valid values are from `60` to `10240`. The default value is `4096`.

**NOTE:** Configuring an undersized value can cause the operation to fail; however, configuring an oversized value causes the size to be larger than necessary. It is recommended that this value is not modified unless that timestamp server requires a response size to be less than 4096 bytes.

**Send Nonce**

Select this option to send a nonce with the request. A *nonce* is a parameter that varies with time. These parameters can be a timestamp, a visit counter on a web page, or a special marker. The parameter is intended to limit or prevent the unauthorized replay or reproduction of a file. When the option deselected, a nonce is not sent with the request. By default, the option is selected.

# 21.158. UpdateVersionType

A `string` that represents how to modify the version number of a document when it is stored in either an EMC Documentum or IBM FileNet repository.

## Literal value

For Documentum, these values are valid:

**Keep Same Version:**

The version number remains the same.

**Increment Major Version:**

Updates the version number with a major version increment.

**Increment Minor Version:**

Updates the version number with a minor version increment.

For FileNet, these values are valid:

**Increment Major Version:**

Updates the version number with a major version increment.

**Increment Minor Version:**

Updates the version number with a minor version increment.

# 21.159. URLSpec

A complex data type that you can use as the value for the URL Options properties of the renderFormGuide, renderHTMLForm(deprecated), renderPDFForm, and operations that the Forms service provides. The variable holds URI values that are required by the Forms service when rendering forms.

For information about data that can be accessed using Xpath Expressions, see Dataitems.

For information about configuring default properties, see Datatypespecific settings.

## Data items

The data items that `URLSpec` variables contain.

### applicationWebRoot

A `string` value that represents the root of the URL that is used to access application-specific web content. This value is combined with the value of the targetURL property to construct an absolute URL. (See targetURL.)

### baseURL

A `string` value that represents the URL that is the HTTP-equivalent of the content root URI and is required for HTML transformations that include HREF references to external dependencies, such as images or scripts. When a dependency path is absolute, this value is ignored. This data item is only applicable for the renderHTMLForm(deprecated) and renderHTMLForm operations.

### contentRootURI

A `string` value that represents the URI or an absolute reference to a location from which forms are retrieved. This value is combined with the value of the Form To Render property of the operations that the variable is used with.

This value can reference these sources:

**Repository:**

The repository contains assets that you upload to AEM Forms Server. The value `repository:///` references the root of the repository. The first two forward slashes are part of the protocol (repository://) and the third forward slash represents the root of the repository.

**Directory in the file system of the AEM Forms Server:**

A location on AEM Forms Server, such as C:*[foldername]*. Using a location on the server is not recommended if you want to maximize the portability of the application.

**Network directory:**

A location on the network such as *[folder name].*

**Web location that is accessible by using HTTP:**

After uploading a file to a location on a web server, you can specify the location using an URL, such as http://*[server name]*:*[port number]*/*[folder name]*.

**options**

A `map` value that contains the options for the renderFormGuide, renderHTMLForm(deprecated), and renderPDFForm operations of the Forms service. The key is the option name.

**targetURL**

A `string` value that represents the URL to a web service or Java servlet that receives data from the client software when a user submits the form. If targetURL is not an absolute URL, it is combined with the value of applicationWebRoot to construct an absolute URL. (See applicationWebRoot.)

The application endpoint that the URL identifies must provide an implementation, such as a Java servlet, that processes the posted data.

## Datatype specific settings

Properties for setting the default values of the variable data.

**Application Web Root**

Sets the URL representing the root location that is used to access application-specific web content. This value is combined with the value of the Target URL option to construct an absolute submit URL.

**Target URL**

Sets the URL to access a web service or Java servlet that receives data from the client application when a user submits the form. Setting a value in this option sets the target URL in the form design to the value specified by this property. If this option is not an absolute URL, it is combined with the value from the Application Web Root option to construct an absolute URL.

**Content Root URI**

Sets the URI or an absolute reference, which specifies the location in the repository to retrieve a form design. This value is combined with the value of the Form To Render property in this operation to construct an absolute path to the form. You can use these sources for a URI or absolute reference:

**Repository:**

The repository contains assets that you upload to AEM Forms Server. The value `repository:///` references the root of the repository. The first two forward slashes are part of the protocol (repository://) and the third forward slash represents the root of the repository. For example, the documents folder is created below the root of the repository.

**Directory in the file system of the AEM Forms Server:**

A folder on AEM Forms Server, such as C:*[foldername]*. Using a location on the server is not recommended if you want to ensure portability of your application.

**Network directory:**

A folder on the network.

**Web location that is accessible by using HTTP:**

A folder accessible using an URL.

**Base URL**

Sets the URL, which is the HTTP-equivalent of the Content Root URI. This value is required only when you render HTML forms (renderHTMLForm(deprecated) and renderHTMLForm operations) that include HREF references to external dependencies, such as images or scripts. When a dependency path is absolute, this value is ignored.

# 21.160. User

A complex data type that represents a user account from User Manager. This variable type is used to store the results of the FindUser and FindUsers operation that the User Lookup service provides. You can also use XPath expressions that resolve to `User` variables as values for the Initial User Selection property of AssignTask operation operations that the User service provides.

`User` variables contain attribute values for user accounts that exist in User Manager.

For information about data that can be accessed using Xpath Expressions, see Dataitems.

## Data items

The data items that `User` variables contain.

### canonicalName

A `string` value that represents the canonical name for the user account.

### commonName

A `string` value that represents the common name of the user account, such as `Tony Blue`.

### domainName

A `string` value that represents the name of the User Manager domain that the user account belongs to.

### email

A `string` value that represents the email address of the user account.

### familyName

A `string` value that represents the surname of the user account, such as `Blue`.

### givenName

A `string` value that represents the first name of the user account, such as `Tony`.

### initials

A `string` value that represents the initials of the user account.

### org

A `string` value that represents the organization that the user account belongs to.

### postalAddress

A `string` value that represents the postal address of the user account.

### principalOid

A `string` value that represents the identifier of the principal object that represents the user account. The principalOid uniquely identifies the user in the AEM Forms environment.

### telephoneNumber

A `string` value that represents the telephone number of the user account.

### userId

A `string` value that represents the identification of the user account. The userId is the user name that is used to log in to AEM Forms.

RELATED LINKS:

Group

User Lookup

Assign Task operation

# 21.161. User Action Name

A complex data type that represents a user action for Process Management tasks. A list of User Action Name values can be used as the value of the User Actions property of AssignTask and AssignMultiple Tasks operations (User 2.0 service).

For information about data that can be accessed using Xpath Expressions, see Dataitems.

## Data items

The data items that User Action Name values contain.

### confirmation

A `boolean` value that indicates whether to display a confirmation message when the user selects the user action:

- A value of true indicates a confirmation message is displayed.
- A value of false indicates no confirmation message is displayed.

### confirmationMessage

A `string` value that represents the confirmation message to display to users when they select the user action.

### destination

A `string` value that represents the name of the next operation to execute when users select the user action. This value is the value of the Name property of the operation. (See Commonoperation properties.)

### destinationId

A `string` value that represents the unique identification of the destination operation.

### name

A `string` value that represents the name of the action as it appears to users. ***NOTE:*** *Do not use commas in action names.*

### routeId

A `string` value that represents the unique identification of the route that leads to the destination operation.

### routeName

A `string` value that represents the name of the route that leads to the destination operation. This value is the value of the Name property of the route.

### toolTip

A `string` value that represents the text that appears when users pause their mouse pointer over the action button in Workspace.

### userActionName

A `string` value that represents the name of the action. This value is equivalent to the name data item. userActionName provides backward-compatibility support.

RELATED LINKS:

Providing actions for submitting tasks

## 21.162. Userlist

A complex data type used to specify a list of principals to add to a policy. `Userlist` values are used to configure Createpolicy from existing policy operations that the Rights Management service provides.

## 21.163. UserPreferenceTO

A complex data type that represents a particular user's preferences. For example, it can hold users' notification settings and searches they have saved. Used with the Review, Commenting, and Approval building block, which is installed with the Managed Review & Approval Solution Accelerator.

For information about data that can be accessed using XPath Expressions, see Dataitems.

### Data items

The data items that UserPreferenceTO values contain.

#### notificationEnabled

A `boolean` value that indicates whether e-mail notification is enabled for this user. If `true`, notification is enabled.

#### savedSearches

A `list` of ReviewSearchFilterTO values that represent saved searches that are associated with the user.

#### umOid

A `string` value that represents the User Management identifier for the user.

# 21.164. VerificationTime

A `string` value that represents the time to verify the signature in a PDF document.

`VerificationTime` variables are used in the following Signature service operations:

VerifyPDF Signature

VerifyPDF Signature (deprecated)

VerifyXML Signature

These string values are valid:

**CURRENT_TIME:**

The time at which the verification operation is performed.

**SECURE_TIME_ELSE_CURRENT_TIME:**

Secure time is the time specified by a trusted time-stamping authority. If the validation returns an Unknown status, the validation is performed using CURRENT_TIME.

**SIGNING_TIME:**

The time at which the signature was applied as specified by the signer's computer.

If you specify SECURE_TIME_ELSE_CURRENT_TIME and validation returns a status of unknown with a trusted timestamp, the validation is checked using `Current Time`.

For information about configuring default properties, see Datatypespecific settings.

## Datatype specific settings

Sets the time to use for verifying signatures in a PDF document. Select one of these values:

**Signing Time:**

The time that the signature was applied as given by the signer's computer.

**Current Time:**

The time that the verification operation is being carried out.

**Secure Time Else Current Time:**

The time specified by a trusted time-stamping authority.

If you specify Secure Time Else Current Time and validation returns a status of unknown with a trusted timestamp, the validation is checked using Current Time.

# 21.165. xfaForm

**NOTE:** This variable type was deprecated in LiveCycle ES2.

A complex data type that you can use to store XML data in XDP format and the location of a file in the repository.

`xfaForm` variables are useful when used in human-centric processes so that the form can be delivered in one of several formats. The variable stores the location of the form and the XDP form data that is captured with the form.

forms workflow provides several processes that can be used as render and submit services for this variable type. The services you use depend on the type of form that you are using. (See Advancedsettings.)

For information about how to configure the General and Enduser UI Items properties, see Commonvariable properties.

For information about data that can be accessed using Xpath Expressions, see Dataitems.

For information about configuring default properties, see Datatypespecific settings.

For information on configuring the render and submit services, see Advanced settings.

For information on configuring schema settings, see SchemaSettings.

## Data items

The data items that `xfaForm` variables contain.

### data

XML-based information that contains form data. The actual form data is stored several levels below this node. The following XPath expression shows the path to the form field data:

```
/process_data/variable_name/xdp/datasets/data/FSFIELDS_
```

*variable_name* is the name of the `xfaForm` variable

`FSFIELDS_` is a default name used for the root element of the form data when the schema of the form data is unknown. If the form schema is known, the actual root element is shown, followed by data elements that represent form fields. The data for each field is accessible using XPath expressions.

The following XPath expression returns all of the form data:

```
/process_data/variable_name/xdp/datasets/data/*
```

### templateUrl

A `string` value that represents the URL of a form that resides in the repository.

## Datatype specific settings

Properties for specifying the form and initial form data.

### Template Url

The URL of the form that is associated with the form data that this `xfaForm` variable holds. The form resides in the repository. The form must be one of these types:

- Adobe PDF form
- Adobe Acrobat form
- Adobe XML form

### Default Data

(Optional) Form data to use to populate the form if no other data is stored in the variable. This data is stored in the `data` data item.

## Advanced settings

Properties for configuring the render and submit services.

You use the following default services to post-process a form after a user submits it:

SubmitFormGuideES Update 1

SubmitHTMLFormES Update 1

SubmitPDFFormES Update 1

### Render Service

Properties for configuring the processing of form data before it is sent to the web application or servlet that delivers the form to end users to open in client software.

*Enable Render Service*

Select Enable Render Service to render the form before it is sent to the web application or servlet that delivers the form to client applications. The render operation is specified using the Operation Name property. (See ServiceName and Operation Name.)

If this option is not selected, no processing of the data is performed before it is sent to the client.

*Service Name and Operation Name*

The service and operation to use to render the form and form data that the xfaForm holds.

Process Management provides several processes that can be used as render and submit services. The services you use depend on the type of form that you are using.

| Type of form | Render service to use |
|---|---|
| Acrobat PDF form | The following processes that forms workflow provides:<br>DefaultRender ES Update 1<br>RenderPDFFormES Update 1 |

| Type of form | Render service to use |
|---|---|
| Adobe PDF form | The following processes that forms workflow provides:<br>RenderFormGuideES Update 1<br>RenderHTMLFormES Update 1<br>RenderPDFFormES Update 1 |
| Adobe XML form | The following operations that the Forms service provides:<br>renderFormGuide<br>renderHTMLForm(deprecated)<br>renderPDFForm<br>The following processes that forms workflow provides:<br>RenderFormGuideES Update 1<br>RenderHTMLFormES Update 1<br>RenderPDFFormES Update 1 |

*Service Input*

The input properties of the operation that is selected as the render service. The properties that appear depend on the render service that is selected.

The default value of each of the properties is determined by the Assign Task operation from the User service that uses the `xfaForm` variable as its input form.

*Service Output*

The output properties of the operation that is selected as the render service. The properties that appear depend on the render service that is selected.

For information about the input and output properties for the specified render service, see Servicereference.

For information about the default render service, see ReadResource Content.

**Submit Service**

Properties for configuring the submit service, which processes the form data after it is submitted by the end user.

*Enable Submit Service*

Determines whether the form data is processed after the form is submitted. When selected, a submit service is used. When not selected, no processing of the data is performed after it is submitted.

*Service Name and Operation Name*

The service and operation to use to process the form data after it is submitted. There is no default submit service.

forms workflow provides several processes that can be used as render and submit services. The services you use depend on the type of form that you are using.

| Type of form | Render service to use |
|---|---|
| Acrobat PDF form | The Submit PDF Form ES Update1 process that forms workflow provides. |
| Adobe PDF form | The following processes that forms workflow provides:<br>• Submit Form Guide ES Update1<br>*Submit HTML Form ES Update1*<br>*Submit PDF Form ES Update1* |
| Adobe XML form | The processFormSubmission operation that the Forms service provides.<br>The following processes that forms workflow provides:<br>• Submit Form Guide ES Update1<br>*Submit HTML Form ES Update1*<br>*Submit PDF Form ES Update1* |

*Service Input*

The input properties of the operation that is selected as the submit service. The properties that appear depend on the submit service that is selected.

*Service Output*

The output properties of the operation that is selected as the submit service. The properties that appear depend on the submit service that is selected.

For information about the input and output properties for the specified submit service, see Servicereference.

### Schema Settings

The XML schema that the form data conforms to. The schema enables XPath Builder to represent the form data in the process data model at run time. Seeing the form data in the model is useful for creating XPath expressions using XPath Builder.

Click Schema Settings to view the schema or select a schema file from the local file system.

RELATED LINKS:

Configuring XML variables for XDP data

## 21.166. xml

Holds an XML document or a partial XML document. You can specify an XML schema that allows you to see the structure of the XML data in XPath Builder.

### Literal value

Valid XML enclosed in quotation marks.

### Example

```
"<name>
user name
</name>"
```

For information about configuring default properties, see Datatypespecific settings.

### Datatype specific settings

Properties for configuring the xml variable.

**Store Form Data As XDP:**

Select to initialize the variable value with required XML elements for valid XDP data. When the option is selected, XML variables ignore prefixes registered for namespaces and XPath expressions do not require the prefix. When deselected, you must use the registered prefixes in the XPath expression.

**Asset Reference:**

The imported asset that defines the schema and root entity to use for the variable.

**Schema:**

The XML schema to use for the variable.

**Root Entity:**

The entity to use as the root element of the XML value.

RELATED LINKS:

XPath Builder (Process Properties)

Configuring XML variables for XDP data

Displaying the xml data structure in XPath Builder

Adding data nodes to xml variables

## 21.167. XMLFormat

A `string` value that represents an XML format. This variable type is used with the Extractto XML operation that the barcoded forms service provides. These values are valid:

**XDP:**

Convert delimited data to XDP data

**XFDF:**

Convert delimited data to XFDF data.

# 21.168. XMLSignatureVerificationResult

A complex data type used by the VerifyXML Signature operation provided by the Signature service. It contains information about the signature and its validity status.

For information about data that can be accessed using Xpath Expressions, see Dataitems.

## Data items

The data items that `XMLSignatureVerificationResult` variables contain.

### certificateList

A `list` of `byte` values that contains the list of certificates.

### dateSigned

A `date` value that specifies the date on which the PDF document was signed.

### signatureStatus

A `boolean` value that indicates whether the signature that is used to sign a PDF document is valid or not. A value of `true` means that the signature is valid. A value of `false` means that the signature is invalid.

### signerCert

A `list` of `byte` values that represents the certificate of the signer.

### signerName

A `string` value that specifies the name of the signer.

### signerStatus

An `IdentityStatus` value that identifies the status of the signer.

# 21.169. XMPUtilityMetadata

A complex data type that represents the metadata that can be imported into a PDF document or exported from a PDF document:

- Use XMPUtilityMetadata values for the Metadata property of ImportMetadata operations (XMP Utilities service).

- Use XMPUtilityMetadata variables to store the output of ExportMetadata operations (XMP Utilities service).

**NOTE:** When used with the Import Metadata operation, XMPUtilityMetadata data items with no assigned values cause the corresponding PDF metadata values to be removed.

### Data items

The data items that `XMPUtiltityMetadata` variables contain.

**author**

A `string` value that represents the author of the PDF document.

**creator**

A `string` value that represents the creator of the PDF document.

**keywords**

A `list` of `string` values that represent the document keywords.

**producer**

A `string` value that represents the producer of the PDF document.

**subject**

A `string` value that represents the subject of the PDF document.

**title**

A `string` value that represents the title of the PDF document.

## 21.170. Find Type

Provides a Search Data Type dialog box that you can use to search for the names of Java classes that the services define. You can create a variable that holds data that any of these Java classes represent.

For example, a Java object that has no corresponding variable type can be passed into the process when it is invoked. You can use the Search Data Type dialog box to create a variable that is suitable for storing the object.

In the Search box, type part or all of the Java class, and then click Search to find matching classes.

**TIP:** To see all of the defined classes, provide no value in the Search box, and click Search.

# 22. Service reference

This content provides reference information about the core service operations for AEM Forms.

For information about the availability of each service within your organization, see Services Referencefor AEM Forms or consult with your administrator.

Each available service may also come with pre-deployed processes that are activated as services. These processes are implementations of product features and should not be altered.

## 22.1. Configuring for proxy servers

If your processes use a service that connects to a server through a proxy server, configure Workbench and the AEM Forms Server so that the connection is successful. For example, if your network is configured such that you must use a proxy server to connect to a server that hosts a web service, configure Workbench and the AEM Forms Server.

Also, when you configure the connection properties of a service or service operation, use connection parameters, such as service URLs, that the proxy server is configured to recognize. When the proxy server receives the service request, it forwards the service request to the correct server.

For example, the Invoke Web Service operation that the Web Service service provides requires that you specify the URL of the web service WSDL. The value of the WSDL URL property must be a value that the proxy server recognizes as a service request to pass on to another server.

For details about connection information to use with your proxy servers, contact the administrator of your proxy server.

### Configuring Workbench

To configure Workbench, add the following lines to the workbench.ini file:

```
-Dhttp.proxyHost=[name of proxy server]
-Dhttp.proxyPort=[port used by proxy server]
-Dhttp.nonProxyHosts=[Names of servers to connect to directly]
```

If you are connecting to multiple servers through the proxy server, separate each server name with the pipe character (|).

For example, the server named *service1* must be connected through the proxy server named *proxy1* by using port 80. However, the servers on the foo.com domain and localhost can be connected to directly, without using the proxy server. The following lines must be added to the workbench.ini file:

```
-Dhttp.proxyHost=proxy1
-Dhttp.proxyPort=80
-Dhttp.nonProxyHosts=*.foo.com|localhost
```

The workbench.ini file is located in the *[install directory]*/AEM Forms Workbench/workbench directory, where *[install directory]* is the location where you installed Workbench.

## Modify the workbench.ini file:

1) Open the workbench.ini file in a text editor.
2) Add the lines that are required for configuring the use of the proxy server.
3) Save the workbench.ini file and restart Workbench.

## Configuring the AEM Forms Server

To configure the AEM Forms Server, configure the following options for Java Virtual Machine (JVM) that runs the server:

**-Dhttp.proxyHost:**

Indicates the name of the proxy server

**-Dhttp.proxyPort:**

Indicates the port that the proxy server uses

**-Dhttp.nonProxyHosts:**

A list of servers that can be connected to directly, without using the proxy server

For example, the server named *service1* must be connected through the proxy server named *proxy1* by using port 80. The servers on the foo.com domain and localhost services can be connected to directly, without using the proxy server. To configure a JBoss Application Server that runs the AEM Forms Server, the following line must be added to the run.bat file:

```
set JAVA_OPTS=%JAVA_OPTS% -Dhttp.proxyHost=proxy1 -Dhttp.proxyPort=80
-Dhttp.nonProxyHosts="*.foo.com|localhost"
```

The following table lists links to AEM forms documentation that provide tips about how to configure JVM options on your application server.

| Application server | Documentation |
|---|---|
| JBoss | See the "Modifying the JBoss configuration files" topic in the Preparing To Install AEMforms guide. |
| WebSphere | See the "Configuring the JVM arguments and properties" topic in the Installing and DeployingAEM Forms for WebSphere guide. |
| WebLogic | See the "Configuring the JVM arguments" topic in the Installing and Deploying AEMforms for WebLogic guide. |

### Digital signatures and proxy servers

In some AEM forms environments, users can access the AEM Forms Server either by using a proxy server or by accessing the server directly. For processes that use the Signature service, all of the users must access the AEM Forms Server in the same way: either directly or through the same proxy server.

For example, an external user digitally signs a PDF form and submits it to the proxy server. The process sends the form to an internal user who connects to the AEM Forms Server directly. When the internal user opens the form, the digital signature is invalidated and an error occurs when the user tries to submit the form.

## 22.2. About deprecated operations

Some service operations are deprecated. Deprecated operations are supported for two major releases after the release in which they are deprecated. Typically, new operations are created to replace and enhance the functionality that the deprecated ones provided.

If you are creating new processes, use the new operations instead of the deprecated ones.

If you upgraded to this release of AEM Forms, your existing applications that use deprecated operations continue to function. Because they will not work after two major releases, consider changing them to use the new operations to take advantage of prolonged support and feature enhancements.

## 22.3. Common operation properties

All service operations have General and RouteEvaluation properties that you can configure. The information is provided here instead of in each operation section in the reference.

### General

General properties describe the operation.

#### Name

The name of the operation is used to identify the operation in several contexts:

- Identifies the operation on the process diagram. Names that are appropriate for the context that the operation is used in make the information on the process diagram easy to understand.
- Identifies the operation in administration console when administrators are viewing process status information at run time.
- Identifies the operation on Business Activity Monitoring (BAM Dashboard) when users are analyzing run-time process data.

### Description

An explanation of how the operation is being used in the process. This information is useful for other process developers when you are working in a shared development environment.

### Category

The name of the logical group in which the service is presented in the Services view. You cannot edit this property value.

### Service Name

The name of the service the operation belongs to. You cannot edit this property value.

### Service Operation

The name of the operation. You cannot edit this property value.

## Route Evaluation

The order in which the routes that follow the operation are evaluated. Configure route evaluation only when multiple routes follow the operation. The route at the top of the list is evaluated first. The process follows the first route that is evaluated as valid.

# 22.4. Process services

A service is created for each process that is activated. The name of the service is the name of the process version that it is created for.

Each process service provides the invoke operation. The input and output variables that are defined in the process determine the input and output properties that must be configured for the operation.

When a process includes the invoke operation of another process, the invoked process is known as a *subprocess*.

## invoke

Invokes the process that the service is associated with.

For information about the General and Route Evaluation property groups, see Commonoperation properties.

### Input

Properties for specifying input values for the invoke operation.

One input property is defined for each process variable that has the Input option selected. The name and data type of the operation property is that of the process variable.

For example, the process includes a `string` variable named `stringVar`. The `stringVar` variable has the Input option selected. The invoke operation that the service for the process provides includes the `stringVar` property in the Input property group.

Input properties are required values if the corresponding process variable has the Required option selected.

**Output**

Properties for specifying locations to store output values for the process.

One output value is returned for each process variable that has the Output option selected. The name and data type of the operation property is that of the process variable.

For example, the process includes a `string` variable named `documentVar`. The `documentVar` variable has the Output property selected. The invoke operation that the service for the process provides includes the `documentVar` property in the Output property group.

Output properties are required values if the corresponding process variable has the Required option selected.

**Invocation Policy**

Determines how the process that uses the invoke operation behaves after invoking the subprocess. The process that uses the invoke operation can either continue to execute after being invoked or wait until the subprocess is complete before proceeding.

**Do Not Wait For Response:**

Select this option to have the invoking process continue executing immediately after the invoke operation starts to execute.

**Wait For Response:**

(Default) Select this option to have the invoking process continue executing only after the invoke operation is completed.

RELATED LINKS:

Invokingsubprocesses

## 22.5. Services not for use in processes

Some processes that implement AEM Forms features are deployed when services that are part of AEM Forms are installed. The services that are created for these processes are not intended to be used in your processes.

**IMPORTANT:** If you modify processes that implement AEM Forms features, the features may no longer function correctly. The modification of these processes is not supported.

The following table lists the processes that implement AEM Forms features.

| Process category | Process name |
|---|---|
| Output | PrintPDFPackage |
| forms workflow | Complete Task<br>Email Notification<br>Queue Sharing<br>Share Tasks For Shared Queues<br>ShareTask Service<br>Workspace Queue Sharing |
| Workbench - Tools | ServerLogViewer |

## 22.6. About Select Asset

The Select Asset dialog box is where you select an asset from an application. Clicking the Local button lists the applications that are on your computer. Clicking the Remote button lists the applications that are in the AEM Forms repository.

### Select an asset:

1) Use one of the following methods to select an asset:
   - In the Enter Or Select An Asset box, type the asset name and path that represents the asset location within the application hierarchy. The path must include the name and version of the application and, optionally, subfolder names. An example of a valid path is /MyApp/1.0/processes.
   - Expand the application tree and select the asset.

## 22.7. Filtering operation properties

For most of the operations, you can view either a full list of properties or a basic list. The basic list shows only the properties that are required. If an operation has an option of switching between the basic and full list, the default choice is basic list.

Use the Basic and All buttons to switch between the views. The buttons are located at the top of the Process Properties view. To view a full list of operation properties, click All. To switch to a basic list of operation properties, click Basic.

## 22.8. Assembler

Enables the manipulation of PDF documents in a process. PDF manipulation is performed according to the commands in a DDX file. (See DDX Reference

.)

For information about using the Assembler service, see Services Reference for AEM forms

.

## Assembler DDX Editor

Use the Assembler DDX Editor dialog box to use DDX elements as a literal value in the operation. You type XML formatted text that contains various DDX elements. DDX elements describe the documents that you want from Assembler. The following is an example of the XML header, DDX root, and a result block that you can type in the Assembler DDX Editor dialog box.

```
<?xml version="1.0" encoding="UTF-8"?>
<DDX xmlns="http://ns.adobe.com/DDX/1.0/" >
<!-- example  of a DDX result block -->
<PDF result="out.pdf">
<PDF source="in1.pdf"/>
</PDF
</DDX>
```

For more information about using DDX elements and DDX commands, see the DDX Reference

.

## Rights Management and Assembler

One possible function of the Assembler service is to combine multiple PDF documents into a single PDF document or PDF Portfolio. You use a DDX file to describe the content of the PDF document or portfolio.

The Rights Management service allows you to lock a PDF document by applying a policy to it. If the DDX file modifies a single PDF document, apply a policy to the document only after it is assembled. If the DDX creates or modifies a PDF Portfolio by adding a locked PDF document as a package file, use a `<PackageFiles source>` element to include the locked PDF document. The `<PackageFiles source>` element is also used to include non-PDF documents in a portfolio.

Ensure that you include the .pdf filename extension in the`<PackageFiles source>`element or a `<File>` child element. Without the filename extension, the PDF viewing application (for example, Adobe Reader) will not open the file.

For more information about creating PDF Portfolios or packages, see the Assembler service topic in Services Reference for AEM forms

and DDX Reference

.

## invokeDDX operation

Executes the DDX file on the specified map of input documents and returns the manipulated PDF documents.

For information about the General and Route Evaluation property groups, see Commonoperation properties.

### Input properties

Properties for specifying the input DDX file and optional documents and environmental values.

*Input DDX Document*

A `document` value that represents the DDX file to be invoked.

If you provide a literal value, the following buttons are available for specifying DDX file contents:

**Ellipsis button :**

Displays the Select Assets dialog box. (See AboutSelect Asset).

**Edit button :**

Displays the Assembler DDX Editor dialog box, in which you can type the DDX elements directly, and then click OK. (See AboutAssembler DDX Editor.) The following is an example of the XML header, DDX root, and a result block.

```
<?xml version="1.0" encoding="UTF-8"?>
<DDX xmlns="http://ns.adobe.com/DDX/1.0/" >

<!-- example  of a DDX result block -->
<PDF result="out.pdf">
<PDF source="in1.pdf"/>
<PDF source="in2.pdf"/>
</PDF
</DDX>
```

You can also use a template value and type DDX elements directly into the dialog box, which may be useful for testing purposes if you pass DDX elements as process variables.

*Input Document Map*

(Optional) A `map` value that represents the documents to be manipulated according to the DDX file. The keys are logical names that are referenced in the DDX file. Each key can reference a `document` value or a `list` value that contains `document` values.

For information about retrieving values from a map or a list, see Accessingdata in data collections.

If you provide a literal value, you can add DDX keys and the document values that each key references. For the DDX keys, the following buttons are available for adding and removing keys.

**Add A List Entry:**

Adds a DDX key to the list.

**Delete Selected List Entry:**

Removes a DDX key from the list.

For each selected DDX key, the following buttons are available for adding, moving, and removing values that represent a document.

**Add A List Entry:**

Adds a document entry to the list.

**Delete Selected List Entry:**

Removes a document entry from the list.

**Move Selected List Entry Up One Row:**

Moves the selected document entry up in the list. The list order determines the order in which document entries are used.

**Move Selected List Entry Down One Row:**

Moves the selected document entry down in the list.

*Environment*

(Optional) An `AssemblerOptionSpec` value that specifies environmental options to be used during processing of the DDX.

If you specify a literal value, specify the following values.

**Job Log Level:**

Sets the log level for this operation. The default is `INFO`. The following values can be used:

- **OFF**: Turn off logging of messages on the server.
- **SEVERE**: Log messages only when they indicate a serious failure.
- **WARNING:**Log messages that indicate potential problems and serious failures.
- **INFO:**Log messages that are for informational purposes, indicate potential problems, and indicate serious failures.
- **CONFIG:**Log only static configuration messages.
- **FINE:**Log tracing information.
- **FINER:**Log fairly detailed tracing information.
- **FINEST:**Log highly detailed tracing information.

**NOTE:** The use of FINE, FINER, and FINEST will negatively impact performance on the server. Use them for debugging purposes only.

**Validate Only:**

Sets whether the DDX job is a validation-only test, which means that the Assembler service should validate the DDX but not execute it. A value of `True` indicates that it is a validation only test. The default is `False`, which indicates that the Assembler service validate and execute the DDX.

**Fail On Error:**

Determines whether the DDX job should exit immediately and stop the processing of any other result blocks when an exception or error occurs. A value of `False` means to keep processing the job regardless of whether an exception or error occurs. Exceptions are accessible as output in the `throwables` data item from the `AssemblerResult` value. The error representing the exception can be seen in the log file in the `jobLog` data item. The default of `True` means that the DDX job should exit immediately, but if your process catches the exception that is thrown, you can access the exception from the `jobLog` data item in the `AssemblerResult` value.

**Default Style:**

The default font style to be set. The font style is the same format as the font attribute for DDX rich text elements: "`<font-style> <font-weight> <font-size>/<line-height> <font-family-specifier>`". If no default is specified, "`normal normal 12pt 'Minion Pro'`"is used.

**First Bates Number:**

The first numeric value to use for Bates numbering. No default is provided.

**Output properties**

Property for specifying the output documents.

If the output includes extracted files attachments (package files), the operation creates unique names for each attachment using the naming scheme of *[filename of source document]*_attach.*[page of attachment]*.*[index of attachment on a page]*:

- *[file name of source document]* is based on the following rules:
  - The file name of the source document. If an URL was provided, the last segment after the last slash in a URL is used. For example, if a URL of http://www.adobe.com/go/ABC.pdf is used, the file name ABC.pdf would be used.
  - If a process URI is used, the portion after the last # character is used for the name. For example, if the process URI `process:///process_data/@doc1#doc1.pdf` is used, the filename doc1.pdf would be used.
  - The document name from the last path segment of the `PDF source` attribute in the DDX file.
- *[page of attachment]* is the page number which the attachment appears in the source document. For example, if an attachment appears on page 7, the value would be 0007.
- *[index of attachment on page]* is of the attachment. This value is important when more than one attachment appears on a page. For example, if there were two attachments that appear on a page, the first one have a value of 0001 and the second attachment would have a value of 0002.

For example, if a source document named PDF123.pdf had two attachments on the first page and 1 attachment on the third page, the names of the attachments would be PDF123.pdf_attach.0001.0001, PDF123.pdf_attach.0001.0002, and PDF123.pdf_attach.0003.0001.

*Assembler Result*

The location to store the returned documents and DDX results. The data type is `AssemblerResult`.

### Exceptions

This operation can throw an OperationException exception.

## invokeDDXOneDocument operation

Executes the DDX file on only one specified document and returns the manipulated PDF document.

For information about the General and Route Evaluation property groups, see Common operation properties.

### Input properties

Properties for specifying the input DDX file, the document to be manipulated using the DDX file, and environmental values.

*Input DDX Document*

A `document` value that represents the DDX file to be invoked. Because there is only one document that can be manipulated using this operation, the logical name for the document is `inDoc`. The `PDF source` attribute in the DDX file must have the value `inDoc`. Here is an example:

```
<PDF source="inDoc"/>
```

If you provide a literal value, the following buttons are available for specifying DDX file contents:

**Ellipsis button :**

Displays the Select Asset dialog box. (See About Select Asset).

**Edit button :**

Displays the Assembler DDX Editor dialog box, in which you can type the DDX elements directly, and then click OK. (See About Assembler DDX Editor.) The following is an example of the XML header, DDX root, and a result block.

```
<?xml version="1.0" encoding="UTF-8"?>
<DDX xmlns="http://ns.adobe.com/DDX/1.0/" >

<!-- example  of a DDX result block -->
<PDF result="out.pdf">
<PDF source="inDoc"/>
</DDX>
```

You can also use a template value and type DDX elements directly into the dialog box, which may be useful for testing purposes if you pass DDX elements as process variables.

*Input Document*

(Optional) A `document` value that represents the document to be manipulated according to the DDX file. The default logical name for the document is `inDoc`.

If you provide a literal value, clicking the ellipsis button displays the Select Asset dialog. (See About Select Asset.)

*Environment*

(Optional) An `AssemblerOptionSpec` value that specifies environmental options to be used during processing of the DDX.

If you provide a literal value, specify the following values:

**Job Log Level:**

Sets the log level for this operation. The default is `INFO`. The following values can be used:

- **OFF**: Turn off logging of messages on the server.
- **SEVERE**: Log messages only when they indicate a serious failure.
- **WARNING:**Log messages that indicate potential problems and serious failures.
- **INFO:**Log messages that are for informational purposes, indicate potential problems, and indicate serious failures.
- **CONFIG:**Log only static configuration messages.
- **FINE:**Log tracing information.
- **FINER:**Log fairly detailed tracing information.
- **FINEST:**Log highly detailed tracing information.

**NOTE:** The use of FINE, FINER, and FINEST will negatively impact performance on the server. Use them for debugging purposes only.

**Validate Only:**

Sets whether the DDX job is a validation-only test, which means that the Assembler service should validate the DDX but not execute it. A value of `True` indicates that it is a validation only test. The default is `False`, which indicates that the Assembler service validate and execute the DDX.

**Fail On Error:**

Determines whether the DDX job should exit immediately and stop the processing of any other result blocks when an exception or error occurs. A value of `False` means to keep processing the job regardless of whether an exception or error occurs. The default of `True` means that the DDX job should exit immediately.

**Default Style:**

The default font style to be set. The font style is the same format as the font attribute for DDX rich text elements: `"<font-style> <font-weight> <font-size>/<line-height> <font-family-specifier>"`. If no default is specified, `"normal normal 12pt 'Minion Pro'"` is used.

**First Bates Number:**

The first numeric value to use for Bates numbering.

**Output properties**

Property for specifying the output document.

*Output Document*

The location to store the returned document. The data type is `document`.

#### Exceptions

This operation can throw an OperationExceptionexception.

### Assembler exceptions

The Assembler service provides the following exceptions for throwing exception events.

#### OperationException

Thrown if an error occurs during the invokeDDX or invokeDDXOneDocument operation.

**NOTE:** To receive an OperationException exception, ensure that the variable defined to contain the exception data is at least 150 characters long.

## 22.9. Asset Placement

The Asset Placement service consumes components of an interactive document request and assembles them into an output document.

The package definition contains a collection of asset references that are "placed" by this service into the output document. A process developer can create interactive statements with variable assets provided through the package definition input parameters.

The output is an interactive PDF Portfolio document that contains a customized portfolio navigator, and other documents that have been specified as attachments for the portfolio.

The Asset Placement service contains a single operation for creating output.

### Generate Interactive Document operation

The Generate Interactive Document operation accepts a number of mandatory and optional parameters. Only the package definition is required. However if you do not specify the data as input to the service, you must specify it in the package definition.

For information about the General and Route Evaluation property groups, see Common operation properties.

#### Input properties

Properties for specifying the package definition; the PDF Portfolio, the data to merge with the document, and the output document.

*Package definition (Mandatory)*

A document value that specifies the number, location, and layout of advertising content that direct marketing will replace. The package definition is XML-based. It includes references to other resources and attachments that will be placed in the document. You create the package definition using the Asset Placement Plug-in for Flash Builder 4 and then add the asset data.

This value is mandatory.

*Document template*

A document value that represents the template PDF Portfolio (.PDF) or Flex navigator (.NAV) that will be used to provide the interactive portion of the output document. The service distinguishes between the two variations and creates the output accordingly. Alternatively, you can set this value to NULL and specify it in the package definition.

*Print template*

A document value that points to a non-interactive printable PDF form. If provided, the PDF document is rendered and placed inside the document, and updated with current data. The value could also be an xfa template stored within a .XDP file with no external references. (A PDF would be passed for efficiency.)

Alternatively, you can set this value to NULL and specify it in the package definition. When defined in the package definition file, the attachment must use the reserved ID "printDocument" so that it is recognized as the printable statement, and merged with statement data.

This value is not mandatory.

*XML data*

A document value that specifies the XML data that must be merged into the document. The data is reflected in the navigator user interface through the flex components, as well as merged into the Print Template if it has been provided.

You must either pass the data to the operation as a parameter, or define it within the package definition file (package.xml). When defined in the package definition file, the resource must use the reserved ID "data" so that it is recognized as the interactive document's global data source.

**Caching properties**

Properties for specifying caching.

*Caching ID*

A string value that the service uses to create a unique cache to hold any resources. This value is a unique identifier for a particular batch run. The intended usage is for operation in batches where many invocations may occur for the same generation of documents. These resources would be shared for the duration. When empty no caching is performed. The Batch Processor service automatically passes this value.

*Clear cache*

Forces the service to clear any cache found with the provided caching ID. This property is useful when an invocation is restarted and you don't want to keep resources.

For efficiency, the service does not clear its cache based on timestamp comparisons between the assets' original location and those it holds in its cache. Rather, by default it holds items while they are being used and expires items as they become idle.

**Output**

Property for specifying the output result.

*Interactive Document result*

A document value that represents the output.

**Exceptions**

This operation can throw an AssetPlacementException.

# 22.10. Barcoded forms

Enables the automatic decoding of data from barcodes located on a form. This service also enables the automatic conversion of data that was extracted from the barcode into XDP or XFDF XML data format.

For information about using the barcoded forms service, see AEM Forms

.

## Decode operation

Decodes all the barcodes in a document and returns another XML document that contains data that was retrieved from the barcodes.

**NOTE:** You cannot decode barcodes in a document that is a dynamic form that was filled in Acrobat and saved as a PDF file.

For information about the General and Route Evaluation property groups, see Commonoperation properties.

**Input properties**

*Document*

A `document` value that represents a PDF document or a TIFF image that contains one or more barcodes to decode.

*PDF 417*

A `boolean` value that, when set to `True`, decodes PDF417 barcodes.

*Data Matrix*

A `boolean` value that, when set to `True`, decodes Data Matrix barcodes.

*QR Code*

A `boolean` value that, when set to `True`, decodes QR Code barcodes.

*Codabar*

A `boolean` value, that when set to `True`, decodes Codabar barcodes.

*Code 128*

A `boolean` value that, when set to `True`, decodes Code 128 barcodes.

*Code 3 of 9*

A `boolean` value that, when set to `True`, decodes Code 39 barcodes.

*EAN13*

A `boolean` value that, when set to `True`, decodes EAN-13 barcodes.

*EAN8*

A `boolean` value that, when set to `True`, decodes EAN-8 barcodes.

*CharSet*

A `Character Set (BarcodedForms)` value that represents the character set encoding value used in the barcode. The specified encoding value must correspond with that used in the barcode.

The default value is `UTF-8`. The following values are valid:

- Big-Five
- GB-2312
- ISO-8859-1
- ISO-8859-2
- ISO-8859-7
- KSC-5601
- Shift-JIS
- UTF-16
- UTF-8
- Hex

  **NOTE:** When Hex is selected, the raw byte data in the barcode is output as a Hex string in the `<xb:content>` element in the output XML. For example, if the barcode contains the UTF-8 characters "ABC", the data in the XML appears as 414243, where 41 is the hex representation of 'A', 42

is the hex representation of 'B', and 43 is the hex representation of 'C'. You can include application logic to convert this Hex value to the original content.

**Output properties**

Property to specify output data.

*OutXMLDoc*

XML that contains all the barcode data that was retrieved from the barcodes. The data type is `org.w3c.dom.Document`.

**Exceptions**

This operation can throw a DecodingException exception.

## Extract to XML operation

Converts delimited barcode data into XDP or XFDF XML data. This allows data that was retrieved from barcodes to be used in other service operations that require XDP or XFDF XML data.

Most implementations use delimited formats to provide high density in the barcode. This operation provides a means to convert from delimited format to XML format.

For information about the General and Route Evaluation property groups, see Commonoperation properties.

**Input properties**

Properties for specifying the barcode data.

*Decoder XML Output*

A `document` value that contains delimited XML barcode data from the Decode operation.

*Line Delimiter*

A `Delimiter` value that represents the type of line delimiter that is used in the document containing barcode data. The line delimiter is the end-of-line character used to separate the field names line from the field values line.

The following values are valid:

- Comma
- Carriage_Return
- Line_Feed
- Tab
- Pipe
- Space
- Semi_Colon

- Colon

  *The default is* `Carriage_Return`*.*

*Field Delimiter*

A `Delimiter` value that represents the type of field delimiter that is used in the document that contains barcode data. The field delimiter is used to separate field names and values.

The following values are valid:

- Comma
- Carriage_Return
- Line_Feed
- Tab
- Pipe
- Space
- Semi_Colon
- Colon

  *The default is* `Tab`*.*

  **NOTE:** The form design should use different values for the line delimiter and field delimiter.

*XML Format*

An `XMLFormat` value that specifies whether to convert the barcode data into XML Data Package (XDP) or XML File Data Format (XFDF), which are Adobe-defined formats. The following values are valid:

- `XDP`
- `XFDF`

  *The default is* `XDP`*.*

**Output properties**

Property to specify the extracted XML.

*OutXMLDocs*

A `list` value that contains values for each barcode. Each value consists of an XML document.

For information about retrieving values from a list, see Accessingdata in data collections.

**Exceptions**

None.

## barcoded forms exceptions

The barcoded forms service provides the following exceptions for throwing exception events.

**DecodingException**

Thrown if an error occurs during a Decode operation.

# 22.11. Batch Processor

The Batch Processor service implements batch processing. Each operation can implement a different type of input and output for the batch items.

The standard operations in the Batch Processor service contain these general groups of properties:

**Job identification**

Identifies a particular batch job instance.

**Service identification**

Specifies the service name or operation name of the service is invoked for each input record.

**Input and output properties**

Defines the input (and optional output). Examples of input and output properties are a file name and a JDBC datasource identified by a JNDI name.

**Optimization properties**

Optimize performance; for example, partitioning strategy and chunk size.

## Get Batch Job operation

Retrieves a batch job.

For information about the General and Route Evaluation property groups, see Common operation properties.

**Input properties**

Property that identifies the batch job to retrieve.

*Job Identifier*

A literal value that identifies the ID of the batch job to retrieve.

**Output property**

Property that identifies the result for the operation.

*Result*

Returns a batch job.

### Exceptions

This operation can throw a BatchProcessorException.

## List Batch Jobs operation

Retrieves a list of batch jobs that represent current and past batch job executions.

For information about the General and Route Evaluation property groups, see Common operation properties.

### Input properties

Properties that identify the batch jobs to retrieve.

*Start Index*

A literal value that identifies the index of the first batch job to retrieve. The first job index is 0.

*Count*

A literal value that identifies the number of batch jobs to retrieve.

### Output property

Property that identifies the result for the operation.

*Result*

Returns a list of batch jobs.

### Exceptions

This operation can throw a BatchProcessorException.

## List Batch Job Steps operation

Retrieves a list of batch job steps that correspond to a batch job execution.

For information about the General and Route Evaluation property groups, see Common operation properties.

### Input properties

Property that identifies the batch job steps to retrieve.

*Job Identifier*

A literal value that identifies the batch job for which to retrieve steps.

### Output property

Property that identifies the result for the operation.

*Result*

Returns a list of batch job steps that correspond to the specified batch job execution.

### Exceptions

This operation can throw a BatchProcessorException.

## List Running Batch Jobs operation

Retrieves a list of batch job that represent currently executing batch jobs.

For information about the General and Route Evaluation property groups, see Common operation properties.

### Input properties

Properties that identify the index of the first batch and the quantity of batch jobs to retrieve.

*Start Index*

A literal value that identifies the index of the first batch job to retrieve. The first job index is 0.

*Count*

A literal value that identifies the number of batch jobs to retrieve.

### Output property

Property that identifies the result for the operation.

*Result*

Returns a list of curently executing batch jobs.

### Exceptions

This operation can throw a BatchProcessorException.

## Purge Batch Repository operation

Purges the batch processor repository of job data that is older than a specified number of days.

For information about the General and Route Evaluation property groups, see Common operation properties.

### Input properties

Property that identifies the age at which batch job data is purged from the repository.

*Age in Days*

A literal value that identifies the maximum age in days of batch job data to retain. Enter a value > 5 to avoid purging currently executing batch jobs.

### Exceptions

This operation can throw a BatchProcessorException.

## Restart Batch Job operation

Restarts a batch job that failed or was stopped.

For information about the General and Route Evaluation property groups, see Common operation properties.

### Input properties

Property that identifies the batch job steps to restart.

*Job Identifier*

A literal value that identifies the batch job to restart.

### Output property

Property that identifies the result for the operation.

*Result*

Batch processing return code. Can return any of the following values:

- COMPLETED
- FAILED
- STOPPED
- UNKNOWN
- ALREADY_RUNNING
- RESTART_FAILED
- ALREADY_COMPLETE

### Exceptions

This operation can throw a BatchProcessorException.

## Run Flat File Job operation

Runs a batch job using a comma-delimited text file. Invokes a service that is part of AEM Forms using parameters read from each row of a delimited text file.

For information about the General and Route Evaluation property groups, see Common operation properties.

### Job Identification properties

Property that identifies the job run.

*Run Identifier*

A string value that uniquely identifies this job run among all job runs with the same parameters. If null or empty, a unique value is generated.

### Service Operation properties

Properties that identify the service and operation to invoke for each batch item read.

*Service*

The name of the AEM Forms service or process to invoke for each input batch item.

*Operation*

The operation to invoke for each batch item that is read. When the service is a process, the value is always `invoke`. When the service is a Document Service Component, the value is the name of the service operation.

### Static Service Parameters properties

Property that identifies static service parameters.

*Static Parameters*

A map that contains the parameters that are applied to the AEM Forms service that is called for each item.

### Batch Input

Properties that identify the name of the input file and the names of the service input parameters that correspond to the fields in the input file.

*Input File Name*

The name of the text file that input items will be read from.

The input file is a text file encoded as UTF-8. Each line represents a batch input item. Lines starting with # are treated as comments.

Within each line, input parameter values are delimited by commas. You can use double-quote characters to escape commas or line endings that are part of values. Use double instances of the double-quote character to escape the double-quote character.

The input text is assumed to be encoded as UTF-8 text. Do not edit the input file in a text editor such as Microsoft ® Notepad and then save as UTF-8 text as it saves the file with a byte order mark.

*Input Field Names*

A comma-delimited list of the names of the service input parameters that correspond to fields in the input file.

**Batch Output**

Properties that identify the name of the output file and the names of the service input parameters that correspond to the fields in the output file.

*Output File Name*

The name of the text file where the output items are written. The file contains one line for each output item. Each line consists of a comma-delimited list of the output parameter values written from the AEM Forms service invocation. If null or empty, no output file is generated.

*Output Field Names*

Output parameters are mapped to fields in a line of the output file using this comma-delimited list of names. If left empty, no output file is generated, using this comma-delimited list of parameter names.

**Optimization properties**

Properties that identify the partition strategy, its parameters, and commit interval.

*Partition Strategy*

Determines how the input items will be partitioned for parallel processing. Use any of the following values:

- No Partitioning. Input items are processed sequentially.
- Each partition has a specified number of input items.
- The input items are split into a specified number of partitions.

*Partition Strategy Parameters*

Arguments used by the partitioning strategy. The number of input items per partition or the number partitions, depending on the partition strategy.

*Commit Interval*

The number of items that will be processed before the batch processing state is committed to persistent store. Setting this value too low may use more resources. Setting this value too high may risk transaction time outs. If a failure occurs, the batch processor needs to retry processing those items.

### Result properties

Batch processing return code. Can return any of the following values:

- COMPLETED
- FAILED
- STOPPED
- UNKNOWN
- ALREADY_RUNNING
- RESTART_FAILED
- ALREADY_COMPLETE

### Exceptions

This operation can throw a BatchProcessorException.

## Run JDBC Cursor Job operation

Invokes a service that is part of AEM Forms using parameters read from each record of a JDBC query. A database cursor reads the input items. This approach is often the simplest and most efficient for reading the input items, although not all databases and application server environments support it.

For information about the General and Route Evaluation property groups, see Common operation properties.

### Job Identification properties

Property that identifies the job run.

*Run Identifier*

A string value that uniquely identifies this job run among all job runs with the same parameters. If null or empty, a unique value is generated.

### Service Operation properties

Properties that identify the service and operation to invoke for each batch item read.

*Service*

The name of the AEM Forms service or process that will be invoked for each batch item that is read.

*Operation*

The operation to invoke for each batch item that is read. When the service is a process, the value is always `invoke`. When the service is a Document Service Component, the value is the name of the service operation.

### Static Service Parameters properties

Properties that identify static service parameters.

*Static Parameters*

A map that contains the parameters that are applied to the AEM Forms service that is called for each item.

### Batch Input properties

Properties that identify the input data source and input SQL.

*Input Data Source*

Input items are read from this data source.

*Input SQL*

An SQL statement where each row of the result set is mapped to a batch input item.

### Batch Output properties

Properties that identify the output data source and output SQL.

*Output Data Source*

Output parameters for each batch item can be written to this data source. If null or empty, no output is written at the batch level.

*Output SQL*

An SQL statement that writes output parameters to the output data source. The SQL statement may use named parameters that correspond to input or output parameters, for example, "insert into mytable (c1, c2) value (:param1, :param2)".

### Optimization properties

Properties that identify the partition strategy, its parameters, and commit interval.

*Partition Strategy*

Determines how the input items will be partitioned for parallel processing. Use any of the following values:

- No Partitioning. Input items are processed sequentially
- Each Partition has a specified number of input items
- The input items are split into a specified number of partitions

*Partition Strategy Parameters*

Arguments used by the partitioning strategy. The number of input items per partition or the number partitions, depending on the partition strategy.

*Commit Interval*

The number of items that will be processed before the batch processing state is committed to persistent store. Setting this value too low may use more resources. Setting this value too high may risk transaction time outs. If a failure occurs, the batch processor needs to retry processing those items.

**Result properties**

Batch processing return code. Can return any of the following values:

- COMPLETED
- FAILED
- STOPPED
- UNKNOWN
- ALREADY_RUNNING
- RESTART_FAILED
- ALREADY_COMPLETE

**Exceptions**

This operation can throw a BatchProcessorException.

## Run JDBC Paging Job operation

Invokes a service that is part of AEM Forms using parameters read from each record of a JDBC query. This method generates database queries that read the rows in a page of input items that are processed in one transaction. Since multiple queries are used to retrieve the input items, this operation can be less efficient than the Run JDBC Cursor Job operation. However, this operation can work with some databases and application servers where the Run JDBC Cursor Job operation is not supported.

For information about the General and Route Evaluation property groups, see Common operation properties.

**Job Identification properties**

Property that identifies the job run.

*Run Identifier*

A string value that uniquely identifies this job run among all job runs with the same parameters. If null or empty, a unique value is generated.

### Service Operation properties

Properties that identify the service and operation to invoke for each batch item read.

*Service*

The name of the AEM Forms service or process that will be invoked for each batch item that is read.

*Operation*

The operation to invoke for each batch item that is read. When the service is a process, the value is always `invoke`. When the service is a Document Service Component, the value is the name of the service operation.

### Static Service Parameters properties

Properties that identify static service parameters.

*Static Parameters*

A set of parameters that are applied to each AEM Forms service invocation.

### Batch Input properties

Properties that identify the input data source and the input SELECT, FROM, and WHERE clauses.

*Input Data Source*

The JNDI name of the JDBC data source that input items will be read from.

*Input SELECT Clause*

The SELECT clause part of SQL query string. You can omit the SELECT keyword.

Do not use the column as a sort key if the column has an alias in the SELECT clause. For example, if you use SELECT clause= "column1, column2, column3 as key" and a user enters "column3" or "key" as the sort key, the JDBC Paging job will fail. The user needs to enter "column1" or "column2" as the sort key.

*Input FROM Clause*

The FROM clause part of SQL query string. You can omit the FROM keyword.

*Input WHERE Clause*

The WHERE clause part of SQL query string. You can omit the WHERE keyword.

*Sort Key*

A unique key to use to sort and limit page content.

### Batch Output properties

Properties that identify the output data source and output SQL.

*Output Data Source*

The JNDI name of a JDBC data source that output items are written to. If null or empty, no output is written at the batch level.

*Output SQL*

An SQL statement that writes output parameters to the output data source. The SQL statement may use named parameters that correspond to input or output parameters, for example, "insert into mytable (c1, c2) value (:param1, :param2)".

### Optimization properties

Properties that identify the partition strategy, its parameters, and commit interval.

*Partition Strategy*

Determines how the input items will be partitioned for parallel processing. Use any of the following values:

- No Partitioning. Input items are processed sequentially
- Each Partition has a specified number of input items
- The input items are split into a specified number of partitions

*Partition Strategy Parameters*

Arguments used by the partitioning strategy. The number of input items per partition or the number partitions, depending on the partition strategy.

*Commit Interval*

The number of items that will be processed before the batch processing state is committed to persistent store. Setting this value too low may use more resources. Setting this value too high may risk transaction time outs. If a failure occurs, the batch processor needs to retry processing those items.

### Result properties

Batch processing return code. Can return any of the following values:

- COMPLETED
- FAILED
- STOPPED
- UNKNOWN
- ALREADY_RUNNING
- RESTART_FAILED
- ALREADY_COMPLETE

### Exceptions

This operation can throw a BatchProcessorException.

## Run XML File Job operation

Invokes a service that is part of AEM Forms using parameters read from each record of an XML file.

For information about the General and Route Evaluation property groups, see Common operation properties.

### Job Identification properties

Property that identifies the job run.

*Run Identifier*

A string value that uniquely identifies this job run among all job runs with the same parameters. If null or empty, a unique value is generated.

### Service Operation properties

Properties that identify the service and operation to invoke for each batch item read.

*Service*

The name of the AEM Forms service or process that will be invoked for each batch item that is read.

*Operation*

The operation to invoke for each batch item that is read. When the service is a process, the value is always `invoke`. When the service is a Document Service Component, the value is the name of the service operation.

### Static Service Parameters properties

Properties that identify static service parameters.

*Static Parameters*

A map that contains the parameters that are applied to the AEM Forms service that is called for each item.

### Batch Input properties

Properties that identify the name of the input file and the names of the service input parameters that correspond to the fields in the input file.

*Input File Name*

Input batch items are read from this XML file.

*Input Item Tag Name*

This element name represents the root of each fragment that represents an input batch item.

**Batch Output properties**

Properties that identify the name of the output file and the names of the service input parameters that correspond to the fields in the output file.

*Output File Name*

An XML file that contains the output parameters values written from the AEM Forms service invocation. If null or empty, no output file is generated.

*Output Item Element Name*

The name of the element that contains each batch output item.

*Output Field Names*

Service output parameters matching the names in this (comma-delimited) list of names are included in the batch output.

*Root Tag Name*

The name of the root element in the output XML file.

**Optimization properties**

Properties that identify the partition strategy, its parameters, and commit interval.

*Partition Strategy*

Determines how the input items will be partitioned for parallel processing. Use any of the following values:

- No Partitioning. Input items are processed sequentially
- Each Partition has a specified number of input items
- The input items are split into a specified number of partitions

*Partition Strategy Parameters*

Arguments used by the partitioning strategy. The number of input items per partition or the number partitions, depending on the partition strategy.

*Commit Interval*

The number of input items that will be processed before the transaction is committed. Setting this value too low may use more resources. Setting this value too high may risk transaction time outs. If a failure occurs, the batch processor needs to retry processing those items.

**Result properties**

Batch processing return code. Can return any of the following values:

- COMPLETED
- FAILED
- STOPPED
- UNKNOWN
- ALREADY_RUNNING
- RESTART_FAILED
- ALREADY_COMPLETE

**Exceptions**

This operation can throw a BatchProcessorException.

### Stop Batch Job operation

Sends a message to a running batch job to stop processing.

For information about the General and Route Evaluation property groups, see Common operation properties.

**Input properties**

Property that identifies the batch job to stop.

*Job Identifier*

A literal value that identifies the batch job to stop.

**Output property**

Property that identifies the result for the operation.

*Stop Message Sent*

Returns a boolean value. Value is true if a request to stop the job was sent successfully.

**Exceptions**

This operation can throw a BatchProcessorException.

## 22.12. Central Migration Bridge

The Central Migration Bridge service invokes a subset of Adobe Central Pro Output Server (Central) functionality, which includes the JFMERGE, JFTRANS, and XMLIMPORT commands. Central Migration Bridge service operations allow you to reuse the following Central assets in AEM forms:

- template design (*.ifd)
- output templates (*.mdf)
- data files (*.dat files)
- preamble files (*.pre files)
- data definition files (*.tdf)

For more information, see “Central Migration Bridge Service” in AEM forms

.

**NOTE:** Before you use Central Migration Bridge service operations, configure the service to reference an installation of Central. (See Central Migration Bridge service configuration.).

## Central Migration Bridge service configuration

To use the Central Migration Bridge service, edit service configuration to provide the location of an existing Central installation. (See Editingservice configurations.)

### Central Install Directory

Specifies the location where Central is installed, such as `C:\Program Files\Adobe\Central`.

The Central installation must exist on the same computer where the AEM forms Server is installed.

## centralDataAccess operation

Extracts the `^form`, `^global`, `^field`, and `^job` commands from the specified data file (*.dat) and transforms the commands to XML data.

For example, your application must use information from a DAT file to determine the next steps in the process. You can use the centralDataAccess operation to transform the data file to XML data and store it in a process variable. Then, you can manipulate and access information about the DAT file by using XPath to determine the next steps in the process.

For information about the General and Route Evaluation property groups, see Common operation properties.

### Input properties

Properties for specifying the DAT file and the number of bytes to process from the specified file.

*Data File*

A `document`value that represents the data file (*.dat) to be processed. The data file is a field-nominated file.

If you provide a literal value, clicking the ellipsis button opens the Select Asset dialog box. (See About Select Asset).

*Bytes To Be Processed*

An `int` value that specifies the number of bytes to process from the provided data file (*.dat). The default is 10000. The number of bytes determines the number of `^form`, `^global`, `^field`, and `^job` commands that are extracted from the data file.

**Output properties**

Property for storing the result of the operation.

*Output XML*

The location in the process data model to store the file containing the XML data. The data type is `document`.

**Exceptions**

This operation can throw a Central Migration Bridge exceptions exception.

## centralMerge operation

Merges the template design (*.ifd) or output template (*.mdf) with the provided data file (*.dat) by invoking the JFMERGE command. Use this command to merge the template design or output template with the data file. Use this operation to create files in formats supported by Central, such as, IPL, ZPL, PDF, PS, or PCL file.

For example, you have a data file that contains customer information, such as the name, address, and invoice number. Your application takes the data file, fills in a form defined by the template design, and sends the output to a printer. You can use the centralMerge operation to merge the data file with the template design. The result of the operation is an output file containing forms that are filled in with information about each customer.

For information about the General and Route Evaluation property groups, see Common operation properties.

**Input properties**

Properties for specifying the input template, data file (*.dat) to be merged, preamble file, and printer location to send the output file to. In addition, you can specify a custom jfmerge.ini file and additional command line options for the JFMERGE command.

*Output Template*

A `document` value that represents the output template (*.mdf) file.

If you provide a literal value, clicking the ellipsis button opens the Select Asset dialog box. (See About Select Asset).

*Data File*

A `document` value that represents the data file (*.dat) to be merged with the output template(*.mdf).

*Preamble File*

A `document` value that represents the preamble file to use. The preamble file (*.pre) contains commands or data that the Print Agent executes on the data file contents before merging with an output template (*.mdf).

*Location*

A `string` value that represents the location to send the output file to. When a file location is specified, the file is saved to a location on the AEM forms Server or in a network location. A file location, such as `C:\mergedFiles\newmergefile.pcl` is saved on the AEM forms Server. When a printer location is specified, for a shared queue, use the shared queue name and server name, such as`\\server-name\sharedqueue`. When the printer location is local to the AEM forms Server, you can use just the shared queue name.

*INI File Path*

A `string` value that represents the location on the server or network of a custom jfmerge.ini file, such as C:\custom\customjfmerge.ini. When no value is provided, the default jfmerge.ini file in the Central installation folder is used.

*Other Command Line Options*

A `string` value that represents additional command line options to affect the processing of the JFMERGE command. When you specify multiple command line options, separate each option with a space. If your option contains spaces, enclose the space with double quotation marks. (““). You can specify any command line options other than `-apr`,`-all`,`-z`.

For more information about the JFMERGE command and the available command line options, see Print Agent Reference

.

**Output properties**

Properties for storing the result of the operation.

*Central Result*

The location in the process data model to store the results of the operation. The data type is `CentralResult`.

*Log Document*

The location in the process data model to store the log file that contains entries for all activities that occur when Central is invoked. The log file includes trace, informational, warning, error, and severe messages from the Central Print Agent. The data type is `document`.

*Response Document*

The location in the process data model to store the response file to record an exit status. By default, the name of the response file the that Print Agent creates is jetform.rsp. This file is created in the same location as the Print Agent executable. The data type is `document`.

For information about response files, see “-arx (Response File)” in the Print Agent Reference

.

*Trace Document*

The location in the process data model to store the trace file. The trace facility is commonly used for troubleshooting. However, it can also be used to generate data files that can be used as preamble for subsequent processing. The data type is `document`.

For information about trace files, see “-atf (Trace File Name)” in the Print Agent Reference

.

*Merged Result Document*

The location in the process data model to store the resulting merged PDF or PCL file. The data type is `document`.

*Exceptions*

This operation can throw a Central Migration Bridge exceptions exception.

## centralTransformation operation

Reformats a file into an accepted formatted data file by invoking the JFTRANS command. The formats include field-nominated, fixed record, overlay, or comma-delimited data.

For example, you have an ASCII file containing customer information, such as the name, address, and invoice number. Your application converts the ASCII file to a data file that is formatted for use with the centralMergecentralMerge operation operation. You use the centralTransformation operation to reformat the ASCII file to a field-nominated data file (*.dat).

For information about the General and Route Evaluation property groups, see Common operation properties.

### Input properties

Properties for specifying the input data file and the definition file. In addition, you can specify a custom jftrans.ini file and additional command line options.

*Data File*

A `document` value that represents the data file (*.dat) to be transformed.

If you provide a literal value, clicking the ellipsis button opens the Select Asset dialog box. (See About Select Asset).

*Data Definition File*

A `document` value that represents definition file (*.tdf) describing how to reformat the data file (*.dat). The data file is created using a text editor or Visual Transformation Editor. The data file describes the incoming data file and defines how to write the data to an output file.

*INI File Path*

A `string` value that represents the location of a custom INI file for the JFTRANS command, such as C:\Custom\customjftrans.ini. When no value is provided, the default jtrans.ini file in the Central installation folder is used.

*Other Command Line Options*

A `string` value that represents additional command line options that can be added to affect the processing of the JFTRANS command. When you specify multiple command line options, separate each option with a space. You can specify any command line options other than `-apr`,`-all`,`-z`.

For information about the JFTRANS command and the available command line options, see Developing Data Transfromations

.

**Output properties**

Properties for storing the result of the operation.

*Central Result*

The location in the process data model to store the results of the operation. The data type is `CentralResult`.

*Log Document*

The location in the process data model to store the log file that contains entries for all activities that occur when Central is invoked. The log file includes trace, informational, warning, error, and severe messages from the Central Print Agent. The data type is `document`.

*Response Document*

The location in the process data model to store the response file to record an exit status. By default, the name of the response file the that Print Agent creates is jetform.rsp. This file is created in the same location as the Print Agent executable. The data type is `document`.

*Trace Document*

The location in the process data model to store the trace file. The trace facility is commonly used for troubleshooting. The data type is `document`.

For information about trace files, see “-atf (Trace File Name)” in Print Agent Reference

.

*Transformed Result Document*

The location in the process data model to store the data file (*.dat). The data type is `document`.

**Exceptions**

This operation can throw a Central Migration Bridge exceptions exception.

## centralXMLImport operation

Transforms XML data into a field-nominated data file (*.dat) by invoking the XMLIMPORT command.

For example, you have an XML file containing customer information, such as the name, address, credit card number, and invoice number. Your applications extracts the XML data, transforms it to a data file, fills in a form defined by the output template (*.mdf), and sends the output to a printer. You can use the centralXMLImport operation to extract the XML and transform it into a data file (*.dat).

For information about the General and Route Evaluation property groups, see Common operation properties.

**Input properties**

Properties for specifying the input XML file, custom XCI files, and additional command line options for the XMLIMPORT command.

*Data File*

A `document` value that represents the XML file to transform, such as C:\xmldata.xml.

If you provide a literal value, clicking the ellipsis button opens the Select Asset dialog box. (See About Select Asset).

*XCI File Path*

A `string` value that represents the location of a custom XCI file for the XMLIMPORT command, such as `C:\Custom\customxmlimport.xci`. When no value is provided, the default xmlimport.xci file in the Central installation folder is used. No default value is provided.

*Other Command Line Options*

A `string` value that represents additional command line options that can be added to affect the processing of the XMLIMPORT command. When you specify multiple command line options, separate each option with a space. You can specify any command line options other than `-config`. No default value is provided.

For information about the XMLIMPORT command and the available command line options, see XML Import Agent Reference XML Import Agent Reference

.

### Output properties

Properties for storing the result of the operation.

*Central Result*

The location in the process data model to store the results of the operation. The data type is `CentralResult`.

*Log Document*

The location in the process data model to store the log file that contains entries for all activities that occur when Central is invoked. The log file includes trace, informational, warning, error, and severe messages from the Central Print Agent. The data type is `document`.

*Response Document*

The location in the process data model to store the response file to record an exit status. By default, the name of the response file the that Print Agent creates is jetform.rsp. This file is created in the same location as the Print Agent executable. The data type is `document`.

*Trace Document*

The location in the process data model to store the trace file. The trace facility is commonly used for troubleshooting. The data type is `document`.

*Test Result Document*

The location in the process data model to store the field-nominated data file (*.dat). The data type is `document`.

### Exceptions

This operation can throw a Central Migration Bridge exceptions exception.

## Central Migration Bridge exception

The Central Migration Bridge service provides the following exceptions for throwing exception events.

### CentralMigrationBridgeException

Thrown if an error occurs during a Central Migration Bridge operation for one of the following reasons:

- The configured Central installation directory is not valid.
- An error occurred starting the JFMERGE, JFTRANS, or XMLIMPORT command.
- An unexpected error occurred during the processing of an operation.
- An interruption occurred when processing an operation.

# 22.13. Content Repository Connector for EMC Documentum

Lets you create processes that interact with content that is stored in a Documentum repository. (See EMC Documentum documentation.)

## Content Repository Connector for EMC Documentum service configuration

The following service configuration property can be set for this service. See Editingservice configurations.

**Asset Link Object Default Path:**

The default portion of the path in the Documentum repository for storing the Asset Link object. The actual path consists of the default path and the location of the form template in the AEM forms repository.

For example, if the default path is set to `/AEMforms/ConnectorforEMCDocumentum/AssetLinkObjects` and the form template is stored in the `/Docbase/forms/` folder, the Asset Link object is stored at the following location:

```
/AEMforms/ConnectorforEMCDocumentum/AssetLinkObjects/Docbase/forms/
```

## Create folders operation

Creates a folder in a Documentum repository.

For information about the General and Route Evaluation property groups, see Commonoperation properties.

### Login Settings properties

Login settings for invoking the Content Repository Connector for EMC Documentum service.

*Login Mode*

A `com.adobe.livecycle.emcdocumentumcontentrepositoryconnector.client.type.impl.LoginSettings` value that represents how the operation will be authenticated with Documentum. Valid values are `INVOCATION_CONTEXT` (Use Credentials from process context), `USER_CREDENTIALS` (Use User Credentials), and `TOKEN`(Use Documentum Login Ticket).

If you provide a literal value, select one of the following options:

**Use Credentials from process context:**

Uses the AEM forms User Management credential from the process context for the Documentum authentication.

**Use User Credentials:**

Uses the User Name and Password properties to authenticate with Documentum. Ensure that the user name is valid for the repository that is specified in the Repository Name property in the Relationship Information property group for this operation.

**Use Documentum Login Ticket:**

Uses the Documentum Login Ticket and User Name properties to authenticate with Documentum. This login ticket must be valid in the current Documentum Foundation Class (DFC) session, and the user name must be set to the correct user of the DFC session.

*User Name*

(Optional) A `string` value that specifies the user name to use when this operation is authenticated with Documentum. No default is provided.

This field is used only when the Login Mode property is set to `USER_CREDENTIALS` (Use User Credentials) or `TOKEN`(Use Documentum Login Ticket).

*Password*

(Optional) A `string` value that specifies the password associated with the user name that is specified in the User Name property to use when this operation is authenticated with Documentum. No default is provided.

This field is used only when the Login Mode property is set to `USER_CREDENTIALS` (Use User Credentials).

*Documentum Login Ticket*

(Optional) A `string` value that specifies the value of an existing Documentum login ticket for a valid DFC session. No default is provided.

This field is used only when the Login Mode property is set to `TOKEN`(Use Documentum Login Ticket).

**Folder Creation Settings properties**

Properties to use when creating a folder in the repository.

*Repository Name*

A `string` value that represents the name of the Documentum repository. You can choose from the list of available repositories from your current connection broker defined in administration console. (See Documentum administration help.) No default is provided.

*Folder Path*

A `string` value that indicates the folder path within the repository where you create the folder.

If you provide a literal value, type the path or click Browse to select the folder from the Documentum repository.

When typing the folder path, use the following rules:

- Do not include the repository name in the folder path.
- Use a forward slash (/) to separate folder names, as shown in this example:

  `/[CabinetName]/[FolderPath]/[New Folder Name]`

**NOTE:** Folders that are specified in the folder path that do not exist are created automatically.

**Result properties**

Properties for the created folder.

*LeafFolderObjectId*

The location to store the object ID of the created folder in the folder path. The data type is `string`.

**Exceptions**

This operation can throw a RepositoryException exception.

## Create folders with type and attributes operation

Creates or modifies folders in a Documentum repository. The folders have specified types and attributes.

For information about the General and Route Evaluation property groups, see Commonoperation properties. In addition, this operation provides the following property groups and exceptions:

Login Settings

Folder Creation Settings

Folder Type and Meta-Data

Result

Exceptions

**Login Settings properties**

Login settings for invoking the Content Repository Connector for EMC Documentum service.

*Login Mode*

A `com.adobe.livecycle.emcdocumentumcontentrepositoryconnector.client.type.impl.LoginSettings` value that represents how the operation will be authenticated with Documentum. Valid values are `INVOCATION_CONTEXT` (Use Credentials from process context), `USER_CREDENTIALS` (Use User Credentials), and `TOKEN`(Use Documentum Login Ticket).

If you provide a literal value, select one of the following options:

**Use Credentials from process context:**

Uses the AEM forms User Management credential from the process context for the Documentum authentication.

**Use User Credentials:**

Uses the User Name and Password properties to authenticate with Documentum. Ensure that the user name is valid for the repository that is specified in the Repository Name property in the Relationship Information property group for this operation.

**Use Documentum Login Ticket:**

Uses the Documentum Login Ticket and User Name properties to authenticate with Documentum. This login ticket must be valid in the current Documentum Foundation Class (DFC) session, and the user name must be set to the correct user of the DFC session.

*User Name*

(Optional) A `string` value that specifies the user name to use when this operation is authenticated with Documentum. No default is provided.

This field is used only when the Login Mode property is set to `USER_CREDENTIALS` (Use User Credentials) or `TOKEN`(Use Documentum Login Ticket).

*Password*

(Optional) A `string` value that specifies the password associated with the user name that is specified in the User Name property to use when this operation is authenticated with Documentum. No default is provided.

This field is used only when the Login Mode property is set to `USER_CREDENTIALS` (Use User Credentials).

*Documentum Login Ticket*

(Optional) A `string` value that specifies the value of an existing Documentum login ticket for a valid DFC session. No default is provided.

This field is used only when the Login Mode property is set to `TOKEN`(Use Documentum Login Ticket).

**Folder Creation Settings properties**

Properties to use when creating a folder in the repository.

*Repository Name*

A `string` value that represents the name of the Documentum repository. You can choose from the list of available repositories from your current connection broker defined in administration console. (See AEM forms administration help.) No default is provided.

*Parent Folder Path*

A `string` value that indicates the folder path within the repository where you create the folder. No default is provided.

If you provide a literal value, type the path or click Browse to select the folder from the Documentum repository.

When typing the folder path, use the following rules:

- Do not include the repository name in the folder path.
- Use a forward slash (/) to separate folder names, as shown in this example:
  `/[CabinetName]/[FolderPath]/`

**NOTE:** Folders that are specified in the folder path that do not exist are created automatically.

*Folder Name*

A `string` value that specifies the name of the folder to create or modify.

**Folder Type and Meta-Data properties**

Folder type and attributes to use when creating a folder.

*Folder Type*

(Optional) A `string` value that specifies the type of folder to create. The value specified in this property is used to populate the `map` value in the Attribute Mapping Table property.

If you provide a literal value, type the value that represents the folder type.

**NOTE:** When no value is provided, the *`map`* value in the Attribute Mapping Table property is not populated.

*Attribute Mapping Table*

(Optional) A `map` value for mapping folder type attributes to process variable values.

The attributes are available only when a value is specified in the Folder Type property. The process variables that you map to attributes must be the same data type. For example, if an attribute is a `string`, it must be mapped to a process variable of data type `string`. The name of an attribute is used as the key.

If you provide a literal value, the table shows the name of the attribute in the Name column and the corresponding folder type in the Type column. To map a process variable to an attribute, select the variable from a list in the Value column.

**Result properties**

Properties for the created folder.

*folderID*

The location to store the object ID of the created folder in the folder path. The data type is `string`.

### Exceptions

This operation can throw a RepositoryException exception.

## Create Relationship operation

Creates a relationship between two documents that are present in a Documentum repository.

For information about the General and Route Evaluation property groups, see Commonoperation properties.

### Login Settings properties

Login settings for invoking the Content Repository Connector for EMC Documentum service.

*Login Mode*

A `com.adobe.livecycle.emcdocumentumcontentrepositoryconnector.client.type.impl.LoginSettings` value that represents how the operation will be authenticated with Documentum. Valid values are `INVOCATION_CONTEXT` (Use Credentials from process context), `USER_CREDENTIALS` (Use User Credentials), and `TOKEN`(Use Documentum Login Ticket).

If you provide a literal value, select one of the following options:

**Use Credentials from process context:**

Uses the AEM forms User Management credential from the process context for the Documentum authentication.

**Use User Credentials:**

Uses the User Name and Password properties to authenticate with Documentum. Ensure that the user name is valid for the repository that is specified in the Repository Name property in the Relationship Information property group for this operation.

**Use Documentum Login Ticket:**

Uses the Documentum Login Ticket and User Name properties to authenticate with Documentum. This login ticket must be valid in the current Documentum Foundation Class (DFC) session, and the user name must be set to the correct user of the DFC session.

*User Name*

(Optional) A `string` value that specifies the user name to use when this operation is authenticated with Documentum. No default is provided.

This field is used only when the Login Mode property is set to `USER_CREDENTIALS` (Use User Credentials) or `TOKEN`(Use Documentum Login Ticket).

*Password*

(Optional) A `string` value that specifies the password associated with the user name that is specified in the User Name property to use when this operation is authenticated with Documentum. No default is provided.

This field is used only when the Login Mode property is set to `USER_CREDENTIALS` (Use User Credentials).

*Documentum Login Ticket*

(Optional) A `string` value that specifies the value of an existing Documentum login ticket for a valid DFC session. No default is provided.

This field is used only when the Login Mode property is set to `TOKEN`(Use Documentum Login Ticket).

**Relationship Creation Settings properties**

Specifies the properties for the relationship you want to establish.

*Repository Name*

A `string` value that represents the name of the Documentum repository. You can choose from the list of available repositories from your current connection broker defined in administration console. (See AEM forms administration help.)

*Relationship Type*

A `string` value that specifies the type of relationship to create between the parent document and the child document.

*Parent Document ID Or Path*

A `string` value that indicates either the parent document ID number or the folder location within the repository where the parent document exists.

If you provide a literal value, type the path or the ID, or click Browse to select the folder from the Documentum repository.

When typing the folder path, use the following rules:

- Do not include the repository name in the folder path.
- Use a forward slash (/) to separate folder names, as shown in this example:

  `/[CabinetName]/[FolderPath]/[documentName]`

*Child Document ID Or Path*

A `string` value that indicates either the child document ID number or the folder location within the repository where the child document exists.

If you provide a literal value, type the path or the ID, or click Browse to select the folder from the Documentum repository.

When typing the folder path, use the following rules:

- Do not include the repository name in the folder path.
- Use a forward slash (/) to separate folder names, as shown in this example:

  `/[CabinetName]/[FolderPath]/[documentName]`

*Child Version Label*

(Optional) A `string` value that specifies the version label of the child document. If set, the child document's chronicle ID is used as the child in the relationship.

*Is Permanent Link?*

(Optional) A `boolean` value that indicates whether the parent-child relationship should be maintained across versions of the parent document.

*Description*

(Optional) A `string` value that specifies the text description for the relationship.

**Exceptions**

This operation can throw a RepositoryException exception.

## Delete Content operation

Deletes the specified document or folder from the Documentum repository. If a folder contains subfolders, they are also deleted.

For information about the General and Route Evaluation property groups, see Commonoperation properties.

**Login Settings properties**

Login settings for invoking the Content Repository Connector for EMC Documentum service.

*Login Mode*

A `com.adobe.livecycle.emcdocumentumcontentrepositoryconnector.client.type.impl.LoginSettings` value that represents how the operation will be authenticated with Documentum. Valid values are `INVOCATION_CONTEXT` (Use Credentials from process context), `USER_CREDENTIALS` (Use User Credentials), and `TOKEN`(Use Documentum Login Ticket).

If you provide a literal value, select one of the following options:

**Use Credentials from process context:**

Uses the AEM forms User Management credential from the process context for the Documentum authentication.

**Use User Credentials:**

Uses the User Name and Password properties to authenticate with Documentum. Ensure that the user name is valid for the repository that is specified in the Repository Name property in the Relationship Information property group for this operation.

**Use Documentum Login Ticket:**

Uses the Documentum Login Ticket and User Name properties to authenticate with Documentum. This login ticket must be valid in the current Documentum Foundation Class (DFC) session, and the user name must be set to the correct user of the DFC session.

*User Name*

(Optional) A `string` value that specifies the user name to use when this operation is authenticated with Documentum. No default is provided.

This field is used only when the Login Mode property is set to `USER_CREDENTIALS` (Use User Credentials) or `TOKEN`(Use Documentum Login Ticket).

*Password*

(Optional) A `string` value that specifies the password associated with the user name that is specified in the User Name property to use when this operation is authenticated with Documentum. No default is provided.

This field is used only when the Login Mode property is set to `USER_CREDENTIALS` (Use User Credentials).

*Documentum Login Ticket*

(Optional) A `string` value that specifies the value of an existing Documentum login ticket for a valid DFC session. No default is provided.

This field is used only when the Login Mode property is set to `TOKEN`(Use Documentum Login Ticket).

**Document/Folder Settings properties**

The repository properties for the document or folder to delete.

*Repository Name*

A `string` value that represents the name of the Documentum repository. You can choose from the list of available repositories from your current connection broker defined in administration console. (See Documentum administration help.)

*Document ID Or Path*

A `string` value that indicates either the document ID number or the folder location within the repository of the document you want to delete.

If you provide a literal value, type the path or the ID or click Browse to select the folder from the Documentum repository.

When typing the folder path, use the following rules:

- Do not include the repository name in the folder path.
- Use a forward slash (/) to separate folder names, as shown in this example:

  `/[CabinetName]/[FolderPath]/[documentName]`

**Exceptions**

This operation can throw a RepositoryException exception.

## Execute DQL Query operation

Executes a DQL query provided as an input.

For information about the General and Route Evaluation property groups, see Commonoperation properties.

**Login Settings properties**

Login settings for invoking the Content Repository Connector for EMC Documentum service.

*Login Mode*

A `com.adobe.livecycle.emcdocumentumcontentrepositoryconnector.client.type.impl.LoginSettings` value that represents how the operation will be authenticated with Documentum. Valid values are `INVOCATION_CONTEXT` (Use Credentials from process context), `USER_CREDENTIALS` (Use User Credentials), and `TOKEN`(Use Documentum Login Ticket).

If you provide a literal value, select one of the following options:

**Use Credentials from process context:**

Uses the AEM forms User Management credential from the process context for the Documentum authentication.

**Use User Credentials:**

Uses the User Name and Password properties to authenticate with Documentum. Ensure that the user name is valid for the repository that is specified in the Repository Name property in the Relationship Information property group for this operation.

**Use Documentum Login Ticket:**

Uses the Documentum Login Ticket and User Name properties to authenticate with Documentum. This login ticket must be valid in the current Documentum Foundation Class (DFC) session, and the user name must be set to the correct user of the DFC session.

*User Name*

(Optional) A `string` value that specifies the user name to use when this operation is authenticated with Documentum. No default is provided.

This field is used only when the Login Mode property is set to `USER_CREDENTIALS` (Use User Credentials) or `TOKEN`(Use Documentum Login Ticket).

*Password*

(Optional) A `string` value that specifies the password associated with the user name that is specified in the User Name property to use when this operation is authenticated with Documentum. No default is provided.

This field is used only when the Login Mode property is set to `USER_CREDENTIALS` (Use User Credentials).

*Documentum Login Ticket*

(Optional) A `string` value that specifies the value of an existing Documentum login ticket for a valid DFC session. No default is provided.

This field is used only when the Login Mode property is set to `TOKEN`(Use Documentum Login Ticket).

**Repository Settings properties**

The properties for the Documentum repository.

*Repository Name*

A `string` value that represents the name of the Documentum repository where the query is executed.

You can choose from the list of available repositories from your current connection broker defined in administration console. (See AEM forms administration help.)

**Query Settings properties**

The query to execute.

*DQL Query*

A `string` value that represents the query to execute.

**Result properties**

The property for the result of the Execute DQL Query operation.

*outExecuteDQLResult*

A `list` value that specifies the results of the query.

### Exceptions

This operation can throw a RepositoryException exception.

## Get Linked LC Assets location operation

Retrieves the AEM forms repository location of the form template for the specified form data object in the Documentum repository.

For information about the General and Route Evaluation property groups, see Commonoperation properties.

### Login Settings properties

Login settings for invoking the Content Repository Connector for EMC Documentum service.

*Login Mode*

A `com.adobe.livecycle.emcdocumentumcontentrepositoryconnector.client.type.impl.LoginSettings` value that represents how the operation will be authenticated with Documentum. Valid values are `INVOCATION_CONTEXT` (Use Credentials from process context), `USER_CREDENTIALS` (Use User Credentials), and `TOKEN`(Use Documentum Login Ticket).

If you provide a literal value, select one of the following options:

**Use Credentials from process context:**

Uses the AEM forms User Management credential from the process context for the Documentum authentication.

**Use User Credentials:**

Uses the User Name and Password properties to authenticate with Documentum. Ensure that the user name is valid for the repository that is specified in the Repository Name property in the Relationship Information property group for this operation.

**Use Documentum Login Ticket:**

Uses the Documentum Login Ticket and User Name properties to authenticate with Documentum. This login ticket must be valid in the current Documentum Foundation Class (DFC) session, and the user name must be set to the correct user of the DFC session.

*User Name*

(Optional) A `string` value that specifies the user name to use when this operation is authenticated with Documentum. No default is provided.

This field is used only when the Login Mode property is set to `USER_CREDENTIALS` (Use User Credentials) or `TOKEN`(Use Documentum Login Ticket).

*Password*

(Optional) A `string` value that specifies the password associated with the user name that is specified in the User Name property to use when this operation is authenticated with Documentum. No default is provided.

This field is used only when the Login Mode property is set to `USER_CREDENTIALS` (Use User Credentials).

*Documentum Login Ticket*

(Optional) A `string` value that specifies the value of an existing Documentum login ticket for a valid DFC session. No default is provided.

This field is used only when the Login Mode property is set to `TOKEN`(Use Documentum Login Ticket).

**ECM Object Settings properties**

Repository properties for the form data object.

*Repository Name*

A `string` value that represents the name of the Documentum repository. You can choose from the list of available repositories from your current connection broker defined in administration console. (See AEM forms administration help.)

*Form Data Path*

A `string` value that indicates the location of the form data within the repository.

If you provide a literal value, type the path or click Browse to select the folder from the Documentum repository.

When typing the folder path, use the following rules:

- Do not include the repository name in the folder path.
- Use a forward slash(/) to separate folder names, as shown in this example:
  `/[CabinetName]/[FolderPath]/[documentName]`

**Relationship Type Settings properties**

Properties of the relationship between the form data and the Asset Link object.

*Relationship Between Form Data and Link Asset Object*

(Optional) A `string` value that represents the relationship between the form data and the Asset Link object.

**Results properties**

Properties of the Asset Link object.

*outGetLinkedLCAssetsLocationResult*

A `IGetLinkedLCAssetsLocationResult` value that contains information about the form template path, form template folder path, and the form template name.

*Linked LC Asset's URL*

A `string` value that represents the full path to the form template in the AEM forms repository.

*Linked LC Asset's Folder Path*

A `string` value that represents the path to the folder containing the form template in the AEM forms repository.

*Linked LC Asset's Name*

A `string` value that represents the name of the form template in the AEM forms repository.

**Exceptions**

This operation can throw a RepositoryException exception.

## Get Related operation

Retrieves a list of IDs for all documents that are related to the specified document.

For information about the General and Route Evaluation property groups, see Commonoperation properties.

**Login Settings properties**

Login settings for invoking the Content Repository Connector for EMC Documentum service.

*Login Mode*

A `com.adobe.livecycle.emcdocumentumcontentrepositoryconnector.client.type.impl.LoginSettings` value that represents how the operation will be authenticated with Documentum. Valid values are `INVOCATION_CONTEXT` (Use Credentials from process context), `USER_CREDENTIALS` (Use User Credentials), and `TOKEN`(Use Documentum Login Ticket).

If you provide a literal value, select one of the following options:

**Use Credentials from process context:**

Uses the AEM formsUser Management credential from the process context for the Documentum authentication.

**Use User Credentials:**

Uses the User Name and Password properties to authenticate with Documentum. Ensure that the user name is valid for the repository that is specified in the Repository Name property in the Relationship Information property group for this operation.

**Use Documentum Login Ticket:**

Uses the Documentum Login Ticket and User Name properties to authenticate with Documentum. This login ticket must be valid in the current Documentum Foundation Class (DFC) session, and the user name must be set to the correct user of the DFC session.

*User Name*

(Optional) A `string` value that specifies the user name to use when this operation is authenticated with Documentum. No default is provided.

This field is used only when the Login Mode property is set to `USER_CREDENTIALS` (Use User Credentials) or `TOKEN`(Use Documentum Login Ticket).

*Password*

(Optional) A `string` value that specifies the password associated with the user name that is specified in the User Name property to use when this operation is authenticated with Documentum. No default is provided.

This field is used only when the Login Mode property is set to `USER_CREDENTIALS` (Use User Credentials).

*Documentum Login Ticket*

(Optional) A `string` value that specifies the value of an existing Documentum login ticket for a valid DFC session. No default is provided.

This field is used only when the Login Mode property is set to `TOKEN`(Use Documentum Login Ticket).

**Relationship Creation Settings properties**

Specifies the properties for the relationship you want to establish.

*Repository Name*

A `string` value that represents the name of the Documentum repository. You can choose from the list of available repositories from your current connection broker defined in administration console. (See AEM forms administration help.)

*Relationship Type*

A `string` value that specifies the type of relationship to create between the parent document and the child document.

*Document ID Or Path*

A `string` value that indicates either the document ID number or the folder location within the repository of the document you want to refer to.

If you provide a literal value, type the path or the ID, or click Browse to select the folder from the Documentum repository.

When typing the folder path, use the following rules:

- Do not include the repository name in the folder path.
- Use a forward slash (/) to separate folder names, as shown in this example:

  ```
  /[CabinetName]/[FolderPath]/[documentName]
  ```

*Is Parent In Relationship?*

A `boolean` value that indicates whether to return all documents that participate as children in the relationship (`True`) or all documents that participate as parents in the relationship (`False`).

**Result properties**

Specifies the property for the result of the Get Related operation.

*Get Related Result*

A `list` value that contains the IDs of documents related to the specified document. (See Accessingdata in data collections.)

For information about retrieving values from a list, see Accessingdata in data collections.

**Exceptions**

This operation can throw a RepositoryException exception.

## Retrieve Content operation

Retrieves a document from an EMC Documentum repository.

For information about the General and Route Evaluation property groups, see Commonoperation properties.

**Login Settings properties**

Login settings for invoking the Content Repository Connector for EMC Documentum service.

*Login Mode*

A `com.adobe.livecycle.emcdocumentumcontentrepositoryconnector.client.type.impl.LoginSettings` value that represents how the operation will be authenticated with Docu-

mentum. Valid values are `INVOCATION_CONTEXT` (Use Credentials from process context), `USER_CREDENTIALS` (Use User Credentials), and `TOKEN`(Use Documentum Login Ticket).

If you provide a literal value, select one of the following options:

**Use Credentials from process context:**

Uses the AEM forms User Management credential from the process context for the Documentum authentication.

**Use User Credentials:**

Uses the User Name and Password properties to authenticate with Documentum. Ensure that the user name is valid for the repository that is specified in the Repository Name property in the Relationship Information property group for this operation.

**Use Documentum Login Ticket:**

Uses the Documentum Login Ticket and User Name properties to authenticate with Documentum. This login ticket must be valid in the current Documentum Foundation Class (DFC) session, and the user name must be set to the correct user of the DFC session.

*User Name*

(Optional) A `string` value that specifies the user name to use when this operation is authenticated with Documentum. No default is provided.

This field is used only when the Login Mode property is set to `USER_CREDENTIALS` (Use User Credentials) or `TOKEN`(Use Documentum Login Ticket).

*Password*

(Optional) A `string` value that specifies the password associated with the user name that is specified in the User Name property to use when this operation is authenticated with Documentum. No default is provided.

This field is used only when the Login Mode property is set to `USER_CREDENTIALS` (Use User Credentials).

*Documentum Login Ticket*

(Optional) A `string` value that specifies the value of an existing Documentum login ticket for a valid DFC session. No default is provided.

This field is used only when the Login Mode property is set to `TOKEN`(Use Documentum Login Ticket).

**Document Settings properties**

Specifies the repository properties for the document to retrieve.

*Repository Name*

A `string` value that represents the name of the Documentum repository. You can choose from the list of available repositories from your current connection broker defined in administration console. (See AEM forms administration help.)

*Document ID Or Path*

A `string` value that indicates either the document ID number or the folder location within the repository.

If you provide a literal value, type the path or the ID, or click Browse to select the folder from the Documentum repository.

When typing the folder path, use the following rules:

- Do not include the repository name in the folder path.
- Use a forward slash(/) to separate folder names, as shown in this example:

  `/[CabinetName]/[FolderPath]/[documentName]`

**Meta-Data properties**

Specifies document metadata properties for the retrieved document.

*Document Class Type*

(Optional) A `string` value that represents the Documentum class type to retrieve from the repository. Specify the repository name in the Document Settings property group. The value specified in this property is used to populate the `map` value in the Attribute Mapping Table property.

If you provide a literal value, type the value that represents the document class type.

**NOTE:** When no value is provided, the `map` value in the Attribute Mapping Table property is not populated.

*Attribute Mapping Table*

(Optional) A `map` value for mapping document attributes to process variable values. During document retrieval, the mapped process variables are set with the value of the corresponding attributes from the document. The attributes are available only when a value is specified in the Document Class Type property.

If you provide a literal value, the table shows the name of the attribute in the Name column and the corresponding document class type in the Type column. To map a process variable to an attribute, select the variable from a list in the Value column.

**Results properties**

Specifies the properties for the document retrieved from the Documentum repository.

*Retrieve Content Result*

An `object` value that stores the content properties that are specified in the Results property group.

*Document Contents*

A `document` value that contains the document content.

*Document ID*

A `string` value that indicates the ID for the document.

*Document Version*

A `string` value that specifies the version number of the document.

*Title Of The Document*

A `string` value that indicates the document title.

*Subject For The Document*

A `string` value that specifies the document subject.

*Keywords for the Document*

A `list` value that specifies all keyword strings that are associated with the document. (See Accessing-data in data collections.)

*Document Class Type*

A `string` value that indicates the Documentum class type for the document.

*Document Content Type*

A `string` value that indicates the Documentum content type for the document.

*Name Of The Document Creator*

A `string` value that indicates the user name of the original author of the document.

*Document Creation Date*

A `date` value that indicates when the original version of the document was created.

*Last Document Modifier*

A `string` value that specifies the user name of the last person who changed the document.

*Last Document Modification Date*

A `date` value that indicates when the document was last changed.

*Attribute Name Value Map*

A `map` value that contains values of attributes that are specified during document retrieval. You can also map individual attributes to process variables in the Meta-Data property group.

For information about retrieving values from a map, see Accessingdata in data collections.

**Exceptions**

This operation can throw a RepositoryException exception.

## Set Link To LC Assets operation

Creates an Asset Link Object (ALO) object that establishes a link between objects, files, or data in the AEM forms repository and assets in the Documentum repository. The properties of the ALO object cannot be modified.

For information about the General and Route Evaluation property groups, see Commonoperation properties.

**Login Settings properties**

Login settings for invoking the Content Repository Connector for EMC Documentum service.

*Login Mode*

A `com.adobe.livecycle.emcdocumentumcontentrepositoryconnector.client.type.impl.LoginSettings` value that represents how the operation will be authenticated with Documentum. Valid values are `INVOCATION_CONTEXT` (Use Credentials from process context), `USER_CREDENTIALS` (Use User Credentials), and `TOKEN`(Use Documentum Login Ticket).

If you provide a literal value, select one of the following options:

**Use Credentials from process context:**

Uses the AEM forms User Management credential from the process context for the Documentum authentication.

**Use User Credentials:**

Uses the User Name and Password properties to authenticate with Documentum. Ensure that the user name is valid for the repository that is specified in the Repository Name property in the Relationship Information property group for this operation.

**Use Documentum Login Ticket:**

Uses the Documentum Login Ticket and User Name properties to authenticate with Documentum. This login ticket must be valid in the current Documentum Foundation Class (DFC) session, and the user name must be set to the correct user of the DFC session.

*User Name*

(Optional) A `string` value that specifies the user name to use when this operation is authenticated with Documentum. No default is provided.

This field is used only when the Login Mode property is set to `USER_CREDENTIALS` (Use User Credentials) or `TOKEN`(Use Documentum Login Ticket).

*Password*

(Optional) A `string` value that specifies the password associated with the user name that is specified in the User Name property to use when this operation is authenticated with Documentum. No default is provided.

This field is used only when the Login Mode property is set to `USER_CREDENTIALS` (Use User Credentials).

*Documentum Login Ticket*

(Optional) A `string` value that specifies the value of an existing Documentum login ticket for a valid DFC session. No default is provided.

This field is used only when the Login Mode property is set to `TOKEN`(Use Documentum Login Ticket).

**ECM Object Settings properties**

Properties for the ECM object.

*Repository Name*

A `string` value that represents the name of the Documentum repository. You can choose from the list of available repositories from your current connection broker defined in administration console. (See AEM forms administration help.)

*Form Data Path*

(Optional) A `string` value that indicates the location of the form data within the repository.

If you provide a literal value, type the path or click Browse to select the folder from the Documentum repository.

When typing the folder path, use the following rules:

- Do not include the object store name in the folder path.
- Use a forward slash (/) to separate folder names, as shown in this example:

  ```
  /[CabinetName]/[FolderPath]/[documentName]
  ```

**Form Template Settings properties**

Repository properties for the form template.

*Path of the form template*

A `string` value that indicates the location of the form template in the AEM forms repository.

If you provide a literal value, type the path or click Browse to select the folder from the AEM forms repository.

When typing the folder path, use a forward slash(/) to separate folder names, as shown in this example:

```
/[FolderPath]/[documentName]
```

**Relationship Settings properties**

Properties related to the relationship between the ECM object and the object in the AEM forms repository.

*Name of the Relationship Between Data and Form Template*

(Optional) A `string` value that represents the name of the relationship between the ECM object and the object in the AEM forms repository.

*Repository Description*

(Optional) A `string` value that contains the description of the relationship between the ECM object and the object in the AEM forms repository

**Exceptions**

This operation can throw a RepositoryException exception.

## Store Content operation

Stores a new document or updates an existing document in an EMC Documentum repository.

For information about the General and Route Evaluation property groups, see Commonoperation properties.

**Login Settings properties**

Login settings for invoking the Content Repository Connector for EMC Documentum service.

*Login Mode*

A `com.adobe.livecycle.emcdocumentumcontentrepositoryconnector.client.type.impl.LoginSettings` value that represents how the operation will be authenticated with Documentum. Valid values are `INVOCATION_CONTEXT` (Use Credentials from process context), `USER_CREDENTIALS` (Use User Credentials), and `TOKEN`(Use Documentum Login Ticket).

If you provide a literal value, select one of the following options:

**Use Credentials from process context:**

Uses the AEM forms User Management credential from the process context for the Documentum authentication.

**Use User Credentials:**

Uses the User Name and Password properties to authenticate with Documentum. Ensure that the user name is valid for the repository that is specified in the Repository Name property in the Relationship Information property group for this operation.

**Use Documentum Login Ticket:**

Uses the Documentum Login Ticket and User Name properties to authenticate with Documentum. This login ticket must be valid in the current Documentum Foundation Class (DFC) session, and the user name must be set to the correct user of the DFC session.

*User Name*

(Optional) A `string` value that specifies the user name to use when this operation is authenticated with Documentum. No default is provided.

This field is used only when the Login Mode property is set to `USER_CREDENTIALS` (Use User Credentials) or `TOKEN`(Use Documentum Login Ticket).

*Password*

(Optional) A `string` value that specifies the password associated with the user name that is specified in the User Name property to use when this operation is authenticated with Documentum. No default is provided.

This field is used only when the Login Mode property is set to `USER_CREDENTIALS` (Use User Credentials).

*Documentum Login Ticket*

(Optional) A `string` value that specifies the value of an existing Documentum login ticket for a valid DFC session. No default is provided.

This field is used only when the Login Mode property is set to `TOKEN`(Use Documentum Login Ticket).

**Document Creation Settings properties**

Properties to use when creating a document to store in the repository.

*Repository Name*

A `string` value that represents the name of the Documentum repository. You can choose from the list of available repositories from your current connection broker defined in administration console. (See AEM forms administration help.)

*Document Name*

A `string` value that indicates the name of the document to use when storing content. If a document with the same object name exists in the repository at the path that you specify for the Folder Path property, this operation updates that document object according to the rule that is defined for the Update Version Type property in the Document Content Settings property group for this operation.

*Folder Path*

A `string` value that indicates the folder location within the repository where you want to create the document.

If you provide a literal value, type the path or click Browse to select the folder from the Documentum repository.

When typing the folder path, use the following rules:

- Do not include the repository name in the folder path.
- Use a forward slash (/) to separate folder names, as shown in this example:

  ```
  /[CabinetName]/[FolderPath]/[documentName]
  ```

*Document Class Type*

A `string` value that indicates the Documentum class type for the document.

**NOTE:** This property is used only if a document with the same name does not exist.

*Document Content Type*

A `string` value that indicates the Documentum content type for the document. If the content type does not exist in the repository, this operation creates it.

**NOTE:** This property is used only if a document with the same name does not exist.

**Document Content Settings properties**

Properties that are related to the content of the document to store in the repository.

*Update Version Type*

(Optional) An `UpdateVersionType` value that represents how to increment the document version updates based on the document type. The following values are valid:

**Keep Same Version:**

Version number remains the same.

**Increment Major Version:**

Updates the version number with a major version increment.

**Increment Minor Version:**

Updates the version number with a minor version increment.

The default is `Keep Same Version`.

**NOTE:** This property is used only if a document of the same name exists.

*Title of the Document*

(Optional) A `string` value that indicates the document title.

*Subject for the Document*

(Optional) A `string` value that specifies the document subject.

*Document Contents*

A `document` value that represents the document content.

If you provide a literal value, clicking the ellipsis button ; displays the Select Asset dialog box. (See AboutSelect Asset.)

**NOTE:** This property is mandatory for the operation to run correctly.

*Keywords for the Document*

Optional) A `list` value that specifies all keyword strings that are associated with the document.

If a literal value is used, you can add new keywords by clicking the Add button and, in the Edit Table Item dialog box, type a new keyword and click OK.

**Meta-Data properties**

Metadata properties for the stored document.

*Attribute Mapping Table*

(Optional) Maps the document attributes to process variables in the given table. While creating or updating the document in the Documentum repository, the mapped document attributes are set on the document. If you leave an attribute value blank, the current value of that attribute in the repository is maintained. To clear the value of an attribute, map the attribute to "" (empty string).

**Results properties**

Properties for the created or updated document.

*Store Content Result*

An `object` value that stores the content properties specified in the Results property group.

*Document ID*

A `string` value that indicates the ID for the document.

*Document Version*

A `string` value that specifies the version number of the document.

*Name Of The Document Creator*

A `string` value that indicates the user name of the original author of the document.

*Document Creation Date*

A `date` value that indicates when the original version of the document was created.

*Last Document Modifier*

A `string` value that specifies the user name of the last person who changed the document.

*Last Document Modification Date*

A `date` value that indicates when the document was last changed.

**Exceptions**

This operation can throw a RepositoryException exception.

### Content Repository Connector for EMC Documentum exceptions

The Content Repository Connector for EMC Documentum service provides the following exceptions for throwing exception events.

**RepositoryException**

Thrown if an error occurs while retrieving or storing a document in a Documentum repository (for example, if the value of an operation parameter is `NULL` or if Documentum returns an error).

## 22.14. Content Repository Connector for IBM Content Manager

Lets you create processes that interact with items that are stored in an IBM Content Manager repository. An item can be either a folder or a document.

### Create item operation

Creates an item in an IBM Content Manager repository.

For information about the General and Route Evaluation property groups, see Commonoperation properties.

### Login Settings properties

Properties for specifying the login settings for invoking the Content Repository Connector for IBM Content Manager service.

*Login Mode*

A `LoginMode` (`com.adobe.livecycle.connectorforibmcm.dsc.client.type.LoginMode`) value that represents how the operation will be authenticated with IBM Content Manager. Valid values are `Use Credentials From Process Context` and `Use User Credentials`.

**NOTE:** `Use Credentials From Process Context` works only when both IBM Content Manager and AEM Forms are synchronized to the same LDAP server.

If you provide a literal value, select one of the following options.

**Use Credentials From Process Context:**

Uses the AEM Forms User Management credential from the process context for the IBM Content Manager authentication. The User Name, Data Store, Password, and Connection String properties are not used.

**NOTE:** To invoke a process with Workbench that uses this operation, type the login name in the Username box using the format *`[User Name]`*`@`*`[Data Store]`*`:`*`[Connection String]`*. In this format, *`[User Name]`* identifies the user ID, *`[Data Store]`* is the name of the repository to connect to, and *`[Connection String]`* is the list of options to use to connect to the repository.

**Use User Credentials:**

Uses the User Name and Password properties to authenticate with IBM Content Manager. Ensure that the user name is valid for the repository that is specified in the Data Store property for this operation.

When this value is selected, provide values for the Data Store, User Name, Password, and Connection String properties.

*Data Store*

(Optional) A `string` value that specifies the name of the data store that the user is to connect to.

This field is used only when the Login Mode property is set to `Use User Credentials`.

If you provide a literal value, type the name of the data store.

*User Name*

(Optional) A `string` value that specifies the user name to use when this operation is authenticated with IBM Content Manager No default is provided.

This field is used only when the Login Mode property is set to `Use User Credentials`.

If you provide a literal value, type the user ID.

*Password*

(Optional) A `string` value that specifies the password associated with the user name that is specified in the User Name property. The password is used when this operation is authenticated with IBM Content Manager. No default is provided.

This field is used only when the Login Mode property is set to `Use User Credentials`.

If you provide a literal value, type the password.

*Connection String*

(Optional) A `string` value that specifies additional options to use for connecting to IBM Content Manager, such as `SCHEMA=`*[schemaName]*, where *[schemaName]* is a case-sensitive value that represents a schema.

If you provide a literal value, type the additional connection options.

**Item Creation Settings properties**

Properties for setting the parameters for creating an item in the repository.

*Item Type*

A `string` value that represents the type of item to create in the repository. Item types are created using administration tools on the IBM Content Manager server. The value specified in this property is used to populate the `map` value in the Attributes property.

If you provide a literal value, type the value that represents the item type.

**NOTE:** When no value is provided, no attributes are populated in the *`map`* value in the Attributes property.

*Mime Type*

A `string` value that specifies the MIME type to use when creating an item in the repository. If the item type is a folder, the value in this property is ignored. The supported MIME types are configured on the IBM Content Managerserver.

If you provide a literal value, type the value that represents the MIME type.

*Item Content*

A `document` value that specifies the contents of the item to create in the repository.

If you provide a literal value, clicking the ellipsis button displays the Select Asset dialog box. (See AboutSelect Asset.)

**Meta-Data properties**

Properties for describing the item that is created in the repository.

*Attributes*

A `map` value for mapping attributes of item types to process variable values. The process variables that you map to attributes must be the same data type. For example, if an attribute is a `string`, it must be mapped to a process variable of data type `string`. The name of an attribute is used as the key.

If you provide a literal value, the list of attributes in the table is based on the item you specified in the Item Type property. You can select a process variable from the Value column to assign to each attribute in the Name column.

For example, item type ABC has an attribute named Source. After selecting ABC as the item type in the table, you can map a process variable to assign to the attribute named Source.

**NOTE:** No value is provided for this property when no value exists in the Item Type property.

**Result properties**

Properties to store the results of the operation.

*Document Information*

The location to store the result of the operation, which contains the attributes and contents of creating an item in the repository. The data type is `DocInfo`.

*PID*

The location to store the value that represents the unique identifier (PID) of the item. The PID (Persistent Identifier) is an identifier used by the IBM Content Manager server to uniquely identify all items. The data type is `string`.

*Item Type*

The location to store the value that represents the item type that was created. The data type is `string`.

*Item Version*

The location to store the value that represents the major version of the item that was created in the repository. The first version of an item is 1. The data type is `string`.

*Item Content*

The location to store the item that was created in the repository. The data type is `document`.

*Attributes*

A `map` value of item attributes presented in a name-value format.

**Exceptions**

This operation can throw a RepositoryException exception.

## Create Folder operation

Creates a folder in the IBM Content Manager repository.

For information about the General and Route Evaluation property groups, see Commonoperation properties.

### Login Settings properties

Properties for specifying the login settings for invoking the Content Repository Connector for IBM Content Manager service.

*Login Mode*

A `LoginMode` (`com.adobe.livecycle.connectorforibmcm.dsc.client.type.LoginMode`) value that represents how the operation will be authenticated with IBM Content Manager. Valid values are `Use Credentials From Process Context` and `Use User Credentials`.

**NOTE:** `Use Credentials From Process Context` works only when both IBM Content Manager and AEM Forms are synchronized to the same LDAP server.

If you provide a literal value, select one of the following options.

**Use Credentials From Process Context:**

Uses the AEM Forms User Management credential from the process context for the IBM Content Manager authentication. The User Name, Data Store, Password, and Connection String properties are not used.

**NOTE:** To invoke a process with Workbench that uses this operation, type the login name in the Username box using the format *`[User Name]@[Data Store]:[Connection String]`*. In this format, *`[User Name]`* identifies the user ID, *`[Data Store]`* is the name of the repository to connect to, and *`[Connection String]`* is the list of options to use to connect to the repository.

**Use User Credentials:**

Uses the User Name and Password properties to authenticate with IBM Content Manager. Ensure that the user name is valid for the repository that is specified in the Data Store property for this operation.

When this value is selected, provide values for the Data Store, User Name, Password, and Connection String properties.

*Data Store*

(Optional) A `string` value that specifies the name of the data store that the user is to connect to.

This field is used only when the Login Mode property is set to `Use User Credentials`.

If you provide a literal value, type the name of the data store.

*User Name*

(Optional) A `string` value that specifies the user name to use when this operation is authenticated with IBM Content Manager No default is provided.

This field is used only when the Login Mode property is set to `Use User Credentials`.

If you provide a literal value, type the user ID.

*Password*

(Optional) A `string` value that specifies the password associated with the user name that is specified in the User Name property. The password is used when this operation is authenticated with IBM Content Manager. No default is provided.

This field is used only when the Login Mode property is set to `Use User Credentials`.

If you provide a literal value, type the password.

*Connection String*

(Optional) A `string` value that specifies additional options to use for connecting to IBM Content Manager, such as `SCHEMA=`*[schemaName]*, where *[schemaName]* is a case-sensitive value that represents a schema.

If you provide a literal value, type the additional connection options.

**Folder Creation Settings properties**

Properties for specifying the parameters for the new folder.

*Parent Folder PID*

(Optional) A `string` value that represents the unique identifier of the parent folder. A relationship between the parent folder and the new folder is automatically created.

The PID (Persistent Identifier) is an identifier used by the IBM Content Manager server to uniquely identify all items.

If you provide a literal value, type the value that represents the unique identifier (PID) of the parent folder.

*Folder Type*

A `string` value that represents the type of folder to create. The value specified in this property is used to populate the `map` value in the Attributes property so that you can assign values from process variables to attributes for the folder type.

If you provide a literal value, type the value that represents the folder type.

**NOTE:** When no value is provided, no attributes are populated in the *`map`* value in the Attributes property.

**Meta-Data properties**

Properties to specify the metadata for the created folder.

*Attributes*

(Optional) A `map` value for mapping attributes to process variable values. The attributes are available only when a value is specified in the Folder Type property. The process variables that you map to attributes must be the same data type. For example, if an attribute is a `string`, it must be mapped to a process variable of data type `string`. The name of an attribute is used as the key.

If you provide a literal value, the list of attributes in the table is based on the item you specified in the Folder Type property. You can select a process variable from the Value column to assign to each attribute in the Name column.

**NOTE:** No value is provided for this property when no value exists in the Item Type property.

**Results properties**

Properties to store the results of the operation.

*Folder Information*

A `DocInfo` value that represents the attributes and contents of the created folder.

*PID*

A `string` value that represents the unique identifier (PID) of the folder The PID (Persistent Identifier) is an identifier used by the IBM Content Manager server to uniquely identify all items.

*Folder Type*

A `string` value that represents the type of folder that was created.

*Folder Version*

A `string` value that represents the major version of the folder that was created in the repository. The first version of a folder is 1.

*Attributes*

A `map` value of folder attributes presented in a name-value format.

**Exceptions**

This operation can throw a RepositoryException exception.

## Create Relationship operation

Creates a relationship between two items in an IBM Content Manager repository. Each relationship has a source and a target.

For information about the General and Route Evaluation property groups, see Commonoperation properties.

### Login Settings properties

Properties for specifying the login settings for invoking the Content Repository Connector for IBM Content Manager service.

*Login Mode*

A `LoginMode` (`com.adobe.livecycle.connectorforibmcm.dsc.client.type.LoginMode`) value that represents how the operation will be authenticated with IBM Content Manager. Valid values are `Use Credentials From Process Context` and `Use User Credentials`.

**NOTE:** `Use Credentials From Process Context` works only when both IBM Content Manager and AEM Forms are synchronized to the same LDAP server.

If you provide a literal value, select one of the following options.

**Use Credentials From Process Context:**

Uses the AEM Forms User Management credential from the process context for the IBM Content Manager authentication. The User Name, Data Store, Password, and Connection String properties are not used.

**NOTE:** To invoke a process with Workbench that uses this operation, type the login name in the Username box using the format *`[User Name]`*`@`*`[Data Store]`*`:`*`[Connection String]`*. In this format, *`[User Name]`* identifies the user ID, *`[Data Store]`* is the name of the repository to connect to, and *`[Connection String]`* is the list of options to use to connect to the repository.

**Use User Credentials:**

Uses the User Name and Password properties to authenticate with IBM Content Manager. Ensure that the user name is valid for the repository that is specified in the Data Store property for this operation.

When this value is selected, provide values for the Data Store, User Name, Password, and Connection String properties.

*Data Store*

(Optional) A `string` value that specifies the name of the data store that the user is to connect to.

This field is used only when the Login Mode property is set to `Use User Credentials`.

If you provide a literal value, type the name of the data store.

*User Name*

(Optional) A `string` value that specifies the user name to use when this operation is authenticated with IBM Content Manager No default is provided.

This field is used only when the Login Mode property is set to `Use User Credentials`.

If you provide a literal value, type the user ID.

*Password*

(Optional) A `string` value that specifies the password associated with the user name that is specified in the User Name property. The password is used when this operation is authenticated with IBM Content Manager. No default is provided.

This field is used only when the Login Mode property is set to `Use User Credentials`.

If you provide a literal value, type the password.

*Connection String*

(Optional) A `string` value that specifies additional options to use for connecting to IBM Content Manager, such as `SCHEMA=`*[schemaName]*, where *[schemaName]* is a case-sensitive value that represents a schema.

If you provide a literal value, type the additional connection options.

**Create Relationship Settings properties**

Properties for specifying the parameters for the relationship to create.

*Source PID*

A `string` value that represents the unique identifier of the source item. The PID (Persistent Identifier) is an identifier used by the IBM Content Manager server to uniquely identify all items.

If you provide a literal value, type the value that represents the unique identifier (PID) of the source item

*Target PID*

A `string` value that represents the unique identifier for the target item. The PID (Persistent Identifier) is an identifier used by the IBM Content Manager server to uniquely identify all items.

If you provide a literal value, type the value that represents the PID of the target item.

*RelationType*

A `string` value that specifies the name of the relation type to create. A relation type can be a maximum of 15 characters.

If you provide a literal value, type the value that represents the relation type.

*New Relationship*

A `boolean` value that sets whether to create the relationship type if it does not exist. A value of `True` means to create the relationship type if it does not exist. A default of `False` means not to create the relation type.

If you provide a literal value, select the New Relationship check box to create a relationship type if the value specified in the RelationType property does not exist. Deselect the New Relationship check box to not create a relationship type if it does not exist.

**IMPORTANT:** When you either deselect the New Relationship or set the value to `False`*and then specify a value in the RelationType property that does not exist, an exception is thrown.*

### Result properties

Properties to store the results of the operation.

*Relationship Information*

The location to store the result of the operation. The data type is `RelationInfo`.

*Relationship Type*

The location to store the relationship type that was created from performing the operation. This data type is `string`.

*RelationshipType Description*

The location to store the description of the relationship type. The data type is `string`.

*Source Document Information*

The location to store the document information of the source item in the newly created relationship. The data type is `DocInfo`.

**TIP:** Use XPath expressions and the Set Value service to obtain the data items in the DocInfo value.

*Target Document Information*

The location to store the document information of the target item in the newly created relationship. The data type is `DocInfo`.

### Exceptions

This operation can throw a RepositoryException exception.

RELATED LINKS:

Specifying template expressions

## Delete Item operation

Deletes an item from the IBM Content Manager repository.

For information about the General and Route Evaluation property groups, see Commonoperation properties.

### Login Settings properties

Properties for specifying the login settings for invoking the Content Repository Connector for IBM Content Manager service.

*Login Mode*

A `LoginMode` (`com.adobe.livecycle.connectorforibmcm.dsc.client.type.LoginMode`) value that represents how the operation will be authenticated with IBM Content Manager. Valid values are `Use Credentials From Process Context` and `Use User Credentials`.

**NOTE:** `Use Credentials From Process Context` works only when both IBM Content Manager and AEM Forms are synchronized to the same LDAP server.

If you provide a literal value, select one of the following options.

**Use Credentials From Process Context:**

Uses the AEM Forms User Management credential from the process context for the IBM Content Manager authentication. The User Name, Data Store, Password, and Connection String properties are not used.

**NOTE:** To invoke a process with Workbench that uses this operation, type the login name in the Username box using the format *`[User Name]@[Data Store]:[Connection String]`*. In this format, *`[User Name]`* identifies the user ID, *`[Data Store]`* is the name of the repository to connect to, and *`[Connection String]`* is the list of options to use to connect to the repository.

**Use User Credentials:**

Uses the User Name and Password properties to authenticate with IBM Content Manager. Ensure that the user name is valid for the repository that is specified in the Data Store property for this operation.

When this value is selected, provide values for the Data Store, User Name, Password, and Connection String properties.

*Data Store*

(Optional) A `string` value that specifies the name of the data store that the user is to connect to.

This field is used only when the Login Mode property is set to `Use User Credentials`.

If you provide a literal value, type the name of the data store.

*User Name*

(Optional) A `string` value that specifies the user name to use when this operation is authenticated with IBM Content Manager No default is provided.

This field is used only when the Login Mode property is set to `Use User Credentials`.

If you provide a literal value, type the user ID.

*Password*

(Optional) A `string` value that specifies the password associated with the user name that is specified in the User Name property. The password is used when this operation is authenticated with IBM Content Manager. No default is provided.

This field is used only when the Login Mode property is set to `Use User Credentials`.

If you provide a literal value, type the password.

*Connection String*

(Optional) A `string` value that specifies additional options to use for connecting to IBM Content Manager, such as `SCHEMA=`*[schemaName]*, where *[schemaName]* is a case-sensitive value that represents a schema.

If you provide a literal value, type the additional connection options.

**Delete Item Details properties**

Properties of the item to delete.

*PID*

A `string` value that represents the unique identifier of the item to delete. The PID (Persistent Identifier) is an identifier used by the IBM Content Manager server to uniquely identify all items.

*allVersions*

(Optional) A `boolean` value that specifies whether to delete all versions of the item. A default value to `True` means that all versions of the item are deleted. A value of `False` means that only the specified version are deleted.

If you provide a literal value, select the allVersions check box to indicate to delete all versions of the item. Deselect the allVersions check box to indicate to delete only the specified version of the item.

**Exceptions**

This operation can throw a RepositoryException exception.

## Get Linked LC Assets Location operation

Retrieves the AEM Forms repository location of the form template for the specified form data object in the IBM Content Manager repository.

For information about the General and Route Evaluation property groups, see Commonoperation properties.

**Login Settings properties**

Properties for specifying the login settings for invoking the Content Repository Connector for IBM Content Manager service.

*Login Mode*

A `LoginMode` (`com.adobe.livecycle.connectorforibmcm.dsc.client.type.LoginMode`) value that

represents how the operation will be authenticated with IBM Content Manager. Valid values are `Use Credentials From Process Context` and `Use User Credentials`.

**NOTE:** `Use Credentials From Process Context` works only when both IBM Content Manager and AEM Forms are synchronized to the same LDAP server.

If you provide a literal value, select one of the following options.

**Use Credentials From Process Context:**

Uses the AEM Forms User Management credential from the process context for the IBM Content Manager authentication. The User Name, Data Store, Password, and Connection String properties are not used.

**NOTE:** To invoke a process with Workbench that uses this operation, type the login name in the Username box using the format *`[User Name]`*`@`*`[Data Store]`*`:`*`[Connection String]`*. In this format, *`[User Name]`* identifies the user ID, *`[Data Store]`* is the name of the repository to connect to, and *`[Connection String]`* is the list of options to use to connect to the repository.

**Use User Credentials:**

Uses the User Name and Password properties to authenticate with IBM Content Manager. Ensure that the user name is valid for the repository that is specified in the Data Store property for this operation.

When this value is selected, provide values for the Data Store, User Name, Password, and Connection String properties.

*Data Store*

(Optional) A `string` value that specifies the name of the data store that the user is to connect to.

This field is used only when the Login Mode property is set to `Use User Credentials`.

If you provide a literal value, type the name of the data store.

*User Name*

(Optional) A `string` value that specifies the user name to use when this operation is authenticated with IBM Content Manager No default is provided.

This field is used only when the Login Mode property is set to `Use User Credentials`.

If you provide a literal value, type the user ID.

*Password*

(Optional) A `string` value that specifies the password associated with the user name that is specified in the User Name property. The password is used when this operation is authenticated with IBM Content Manager. No default is provided.

This field is used only when the Login Mode property is set to `Use User Credentials`.

If you provide a literal value, type the password.

*Connection String*

(Optional) A `string` value that specifies additional options to use for connecting to IBM Content Manager, such as `SCHEMA=`*[schemaName]*, where *[schemaName]* is a case-sensitive value that represents a schema.

If you provide a literal value, type the additional connection options.

**Form Data Settings properties**

Repository properties for the form data object.

*Form Data ID*

A `string` value that indicates the location of the form data within the repository.

If you provide a literal value, specify a folder location using the following rules:

- Do not include the repository name in the folder path.
- Use a forward slash(/) to separate folder names if you specify a folder location, as shown in this example:

  `/[CabinetName]/[FolderPath]/[documentName]`

**Relationship Data Settings properties**

Properties of the relationship between the form data and the Asset Link object.

*Relationship Between Form Data And Asset Link Object*

(Optional) A `string` value that represents the relationship between the form data and the Asset Link object.

**Output properties**

Properties of the Asset Link object.

*outGetLinkedLCAssetsLocationResult*

A `IGetLinkedLCAssetsLocationResult` value that contains information about the form template path, form template folder path, and the form template name.

*Linked LC Asset's URL*

A `string` value that represents the full path to the form template in the AEM Forms repository.

*Linked LC Asset's Folder Path*

A `string` value that represents the path to the folder containing the form template in the AEM Forms repository.

*Linked LC Asset's Name*

A `string` value that represents the name of the form template in the AEM Forms repository.

**Exceptions**

This operation can throw a RepositoryException exception.

## Get Related Items operation

When provided an item, this operation retrieves all the related items for it.

For information about the General and Route Evaluation property groups, see Commonoperation properties.

**Login Settings properties**

Properties for specifying the login settings for invoking the Content Repository Connector for IBM Content Manager service.

*Login Mode*

A `LoginMode` (`com.adobe.livecycle.connectorforibmcm.dsc.client.type.LoginMode`) value that represents how the operation will be authenticated with IBM Content Manager. Valid values are `Use Credentials From Process Context` and `Use User Credentials`.

**NOTE:** `Use Credentials From Process Context` works only when both IBM Content Manager and AEM Forms are synchronized to the same LDAP server.

If you provide a literal value, select one of the following options.

**Use Credentials From Process Context:**

Uses the AEM Forms User Management credential from the process context for the IBM Content Manager authentication. The User Name, Data Store, Password, and Connection String properties are not used.

**NOTE:** To invoke a process with Workbench that uses this operation, type the login name in the Username box using the format `[User Name]@[Data Store]:[Connection String]`. In this format, `[User Name]` identifies the user ID, `[Data Store]` is the name of the repository to connect to, and `[Connection String]` is the list of options to use to connect to the repository.

**Use User Credentials:**

Uses the User Name and Password properties to authenticate with IBM Content Manager. Ensure that the user name is valid for the repository that is specified in the Data Store property for this operation.

When this value is selected, provide values for the Data Store, User Name, Password, and Connection String properties.

*Data Store*

(Optional) A `string` value that specifies the name of the data store that the user is to connect to.

This field is used only when the Login Mode property is set to `Use User Credentials`.

If you provide a literal value, type the name of the data store.

*User Name*

(Optional) A `string` value that specifies the user name to use when this operation is authenticated with IBM Content Manager No default is provided.

This field is used only when the Login Mode property is set to `Use User Credentials`.

If you provide a literal value, type the user ID.

*Password*

(Optional) A `string` value that specifies the password associated with the user name that is specified in the User Name property. The password is used when this operation is authenticated with IBM Content Manager. No default is provided.

This field is used only when the Login Mode property is set to `Use User Credentials`.

If you provide a literal value, type the password.

*Connection String*

(Optional) A `string` value that specifies additional options to use for connecting to IBM Content Manager, such as `SCHEMA=`*[schemaName]*, where *[schemaName]* is a case-sensitive value that represents a schema.

If you provide a literal value, type the additional connection options.

**Get Related Items Settings properties**

Properties to specify the item to retrieve related items for.

*PID*

A `string` value that specifies the unique identifier (PID) of the item to retrieve the related items for. The PID (Persistent Identifier) is an identifier used by the IBM Content Manager server to uniquely identify all items.

If you provide a literal value, type the value that represents the PID.

*Relationship Type*

A `string` value that specifies the name of the relation type to retrieve. A relation type can be a maximum of 15 characters.

If you provide a literal value, type the value that represents the relation type.

*Is Source PID*

A `boolean` value that sets whether the value specified in the PID property is the source or target of the relation. A value of `True` means that the value configured in the PID property is the source relation. The default is `False`, which means that the value configured in the PID property is the target relation.

If you provide a literal value, select the Is Source PID check box to specify that the value in the PID property is the source relation. Deselect the Is Source PID check box to specify that the value in the PID property is the target relation. The default for the Is Source PID check box is deselected.

**Results properties**

Properties to specify the result of the operation.

*Related Result*

The location to store the related items that are retrieved from the repository. The data type is a `list` of `DocInfo`.

For information about retrieving values from a list, see Accessingdata in data collections.

**Exceptions**

This operation can throw a RepositoryException exception.

## Retrieve Item operation

Retrieves a document from the IBM Content Manager repository.

For information about the General and Route Evaluation property groups, see Commonoperation properties.

**Login Settings properties**

Properties for specifying the login settings for invoking the Content Repository Connector for IBM Content Manager service.

*Login Mode*

A `LoginMode` (`com.adobe.livecycle.connectorforibmcm.dsc.client.type.LoginMode`) value that represents how the operation will be authenticated with IBM Content Manager. Valid values are `Use Credentials From Process Context` and `Use User Credentials`.

**NOTE:** `Use Credentials From Process Context` works only when both IBM Content Manager and AEM Forms are synchronized to the same LDAP server.

If you provide a literal value, select one of the following options.

**Use Credentials From Process Context:**

Uses the AEM Forms User Management credential from the process context for the IBM Content Manager authentication. The User Name, Data Store, Password, and Connection String properties are not used.

**NOTE:** To invoke a process with Workbench that uses this operation, type the login name in the Username box using the format `[User Name]@[Data Store]:[Connection String]`. In this format, `[User Name]` identifies the user ID, `[Data Store]` is the name of the repository to connect to, and `[Connection String]` is the list of options to use to connect to the repository.

**Use User Credentials:**

Uses the User Name and Password properties to authenticate with IBM Content Manager. Ensure that the user name is valid for the repository that is specified in the Data Store property for this operation.

When this value is selected, provide values for the Data Store, User Name, Password, and Connection String properties.

*Data Store*

(Optional) A `string` value that specifies the name of the data store that the user is to connect to.

This field is used only when the Login Mode property is set to `Use User Credentials`.

If you provide a literal value, type the name of the data store.

*User Name*

(Optional) A `string` value that specifies the user name to use when this operation is authenticated with IBM Content Manager No default is provided.

This field is used only when the Login Mode property is set to `Use User Credentials`.

If you provide a literal value, type the user ID.

*Password*

(Optional) A `string` value that specifies the password associated with the user name that is specified in the User Name property. The password is used when this operation is authenticated with IBM Content Manager. No default is provided.

This field is used only when the Login Mode property is set to `Use User Credentials`.

If you provide a literal value, type the password.

*Connection String*

(Optional) A `string` value that specifies additional options to use for connecting to IBM Content Manager, such as `SCHEMA=`*[schemaName]*, where *[schemaName]* is a case-sensitive value that represents a schema.

If you provide a literal value, type the additional connection options.

### Retrieve Item Settings properties

Property to specify the item to retrieve from the repository.

*PID*

A `string` value that represents the unique identifier of the item to retrieve from the repository. The PID (Persistent Identifier) is an identifier used by the IBM Content Manager server to uniquely identify all items.

If you provide a literal value, type the value that represents the PID.

### Meta-Data properties

Properties to specify the metadata properties for the retrieved document.

*Item Type*

(Optional) A `string` value that represents the item type of the item to retrieve from the repository. The value specified in this property is used to populate the `map` value in the Attributes property so that you can assign values from process variables to attributes for the item type.

If you provide a literal value, type the value that represents the item type.

**NOTE:** When no value is provided, no attributes are populated in the `map` value in the Attributes property.

*Attributes*

A `map` value for mapping attributes to process variable values. The attributes are available only when a value is specified in the Item Type property. The process variables that you map to attributes must be the same data type. For example, if an attribute is a `string`, it must be mapped to a process variable of data type `string`. The name of an attribute is used as the key.

If you provide a literal value, the list of attributes in the table is based on the item you specified in the Item Type property. You can select a process variable from the Value column to assign to each attribute in the Name column.

**NOTE:** No value is provided for this property when no value exists in the Item Type property.

### Results properties

Properties to store the results of the operation.

*Document Information*

The location to store the value that represents the attributes and contents of the item that was retrieved. The data type is `DocInfo`.

*PID*

The location to store the value that represents the unique identifier (PID) of the item retrieved. The PID (Persistent Identifier) is an identifier used by the IBM Content Manager server to uniquely identify all items. The data type is `string`.

*Item Type*

The location to store the value that represents the item type of the item retrieved. The data type is `string`.

*Item Version*

The location to store the value that represents the version for the item retrieved. The data type is `string`.

*Item Content*

The location to store the value that represents the contents of the item retrieved. The data type is `document`.

*Attributes*

The location to store the value that represents the attribute names and values of the item type for the retrieved item. The data type is `map`. (See Accessingdata in data collections.)

**Exceptions**

This operation can throw a RepositoryException exception.

## Search Items operation

Searches for items based on search criteria in an IBM Content Manager repository.

For information about the General and Route Evaluation property groups, see Commonoperation properties.

**Login Settings properties**

Properties for specifying the login settings for invoking the Content Repository Connector for IBM Content Manager service.

*Login Mode*

A `LoginMode` (`com.adobe.livecycle.connectorforibmcm.dsc.client.type.LoginMode`) value that represents how the operation will be authenticated with IBM Content Manager. Valid values are `Use Credentials From Process Context` and `Use User Credentials`.

**NOTE:** `Use Credentials From Process Context` works only when both IBM Content Manager and AEM Forms are synchronized to the same LDAP server.

If you provide a literal value, select one of the following options.

**Use Credentials From Process Context:**

Uses the AEM Forms User Management credential from the process context for the IBM Content Manager authentication. The User Name, Data Store, Password, and Connection String properties are not used.

**NOTE:** To invoke a process with Workbench that uses this operation, type the login name in the Username box using the format `[User Name]@[Data Store]:[Connection String]`. In this format, `[User Name]` identifies the user ID, `[Data Store]` is the name of the repository to connect to, and `[Connection String]` is the list of options to use to connect to the repository.

**Use User Credentials:**

Uses the User Name and Password properties to authenticate with IBM Content Manager. Ensure that the user name is valid for the repository that is specified in the Data Store property for this operation.

When this value is selected, provide values for the Data Store, User Name, Password, and Connection String properties.

*Data Store*

(Optional) A `string` value that specifies the name of the data store that the user is to connect to.

This field is used only when the Login Mode property is set to `Use User Credentials`.

If you provide a literal value, type the name of the data store.

*User Name*

(Optional) A `string` value that specifies the user name to use when this operation is authenticated with IBM Content Manager No default is provided.

This field is used only when the Login Mode property is set to `Use User Credentials`.

If you provide a literal value, type the user ID.

*Password*

(Optional) A `string` value that specifies the password associated with the user name that is specified in the User Name property. The password is used when this operation is authenticated with IBM Content Manager. No default is provided.

This field is used only when the Login Mode property is set to `Use User Credentials`.

If you provide a literal value, type the password.

*Connection String*

(Optional) A `string` value that specifies additional options to use for connecting to IBM Content Manager, such as `SCHEMA=`*[schemaName]*, where *[schemaName]* is a case-sensitive value that represents a schema.

If you provide a literal value, type the additional connection options.

### Search Item Settings properties

Properties to specify the search parameters. The search is performed based on the values provided in the Attributes As Map Of Name-Value Pair property.

*Item Type*

A `string` value that represents the item type of the item to search for in the repository. The value specified in this property is used to populate the `map` value in the Attributes property so that you can assign values from process variables to attributes for the item type.

If you provide a literal value, type the value that represents the item type.

**NOTE:** When no value is provided, no attributes are populated in the `map` value in the property.

*Attributes*

A `map` value for assigning process variable values to an attribute of an item type. The attributes are available only when a value is specified in the Item Type property. The process variables that you map to attributes must be the same data type. For example, if an attribute is of type `string`, it must be mapped to a process variable of data type `string`. The name of an attribute is used as the key.

If you provide a literal value, the list of attributes in the table is based on the item you specified in the Item Type property. You can select a process variable from the Value column to assign to each attribute in the Name column.

**NOTE:** No value is provided for this property when no value exists in the Item Type property.

### Result properties

Property to store the result of the operation.

*List of Items*

The location to store the search results. Only the metadata of each result is retrieved. The data type is a `list` of `DocInfo` values.

For information about retrieving values from a list, see Accessingdata in data collections.

**NOTE:** To retrieve the content for each search result, you can use the RetrieveItem operation and the PID from the `DocInfo` value.

### Exceptions

This operation can throw a RepositoryException exception.

## Set Link To LC Assets operation

Creates an Asset Link Object (ALO) object that establishes a link between objects, files, or data in the AEM Forms repository and assets in the IBM Content Manager repository. The properties of the ALO object cannot be modified.

For information about the General and Route Evaluation property groups, see Commonoperation properties.

### Login Settings

Properties for specifying the login settings for invoking the Content Repository Connector for IBM Content Manager service.

*Login Mode*

A `LoginMode` (`com.adobe.livecycle.connectorforibmcm.dsc.client.type.LoginMode`) value that represents how the operation will be authenticated with IBM Content Manager. Valid values are `Use Credentials From Process Context` and `Use User Credentials`.

**NOTE:** `Use Credentials From Process Context` works only when both IBM Content Manager and AEM Forms are synchronized to the same LDAP server.

If you provide a literal value, select one of the following options.

**Use Credentials From Process Context:**

Uses the AEM Forms User Management credential from the process context for the IBM Content Manager authentication. The User Name, Data Store, Password, and Connection String properties are not used.

**NOTE:** To invoke a process with Workbench that uses this operation, type the login name in the Username box using the format *`[User Name]@[Data Store]:[Connection String]`*. In this format, *`[User Name]`* identifies the user ID, *`[Data Store]`* is the name of the repository to connect to, and *`[Connection String]`* is the list of options to use to connect to the repository.

**Use User Credentials:**

Uses the User Name and Password properties to authenticate with IBM Content Manager. Ensure that the user name is valid for the repository that is specified in the Data Store property for this operation.

When this value is selected, provide values for the Data Store, User Name, Password, and Connection String properties.

*Data Store*

(Optional) A `string` value that specifies the name of the data store that the user is to connect to.

This field is used only when the Login Mode property is set to `Use User Credentials`.

If you provide a literal value, type the name of the data store.

*User Name*

(Optional) A `string` value that specifies the user name to use when this operation is authenticated with IBM Content Manager No default is provided.

This field is used only when the Login Mode property is set to `Use User Credentials`.

If you provide a literal value, type the user ID.

*Password*

(Optional) A `string` value that specifies the password associated with the user name that is specified in the User Name property. The password is used when this operation is authenticated with IBM Content Manager. No default is provided.

This field is used only when the Login Mode property is set to `Use User Credentials`.

If you provide a literal value, type the password.

*Connection String*

(Optional) A `string` value that specifies additional options to use for connecting to IBM Content Manager, such as `SCHEMA=`*[schemaName]*, where *[schemaName]* is a case-sensitive value that represents a schema.

If you provide a literal value, type the additional connection options.

**Form Data and Relationship Settings properties**

Properties of the relationship source object and settings.

*ECMObjectID*

(Optional) A `string` value that specifies the ID of the ECM object.

*Name Of The Relationship Between Data And Form Template*

(Optional) A `string` value that specifies the name of the relationship between the ECM object and the object in the AEM Forms repository.

If you provide a literal value, you can either select a name that exists in the IBM Content Manager server or specify a new name.

**Form Template and ALO Settings properties**

Properties for the form template and the Asset Link object.

*Path of the form template*

A `string` value that indicates the location of the form template in the AEM Forms repository.

If you provide a literal value, type the path or click Browse to select the folder from the AEM Forms repository.

When typing the folder path, use a forward slash(/) to separate folder names if you specify a folder location, as shown in this example:

```
/[FolderPath]/[documentName]
```

*Item Type of the Asset Link Object*

(Optional) A `string` value that represents the item type of the item Asset Link object. The value specified in this property is used to populate the `map` value in the Attributes property so that you can assign values from process variables to attributes for the item type.

If you provide a literal value, type the value that represents the item type.

**NOTE:** When no value is provided, no attributes are populated in the `map` value in the property.

**Meta-Data properties**

The metadata of the Asset Link object.

*Attributes*

A `map` value for assigning process variable values to an attribute of an item type. The attributes are available only when a value is specified in the Item Type Of The Asset Link Object property. The process variables that you map to attributes must be the same data type. For example, if an attribute is of type `string`, it must be mapped to a process variable of data type `string`. The name of an attribute is used as the key.

If you provide a literal value, the list of attributes in the table is based on the item you specified in the Item Type Of The Asset Link Object property. You can select a process variable from the Value column to assign to each attribute in the Name column.

**NOTE:** No value is provided for this property when no value exists in the Item Type Of The Asset Link Object property.

**Exceptions**

This operation can throw a RepositoryException exception.

## Update Item operation

Updates an existing item in an IBM Content Manager repository.

For information about the General and Route Evaluation property groups, see Commonoperation properties.

**Login Settings properties**

Properties for specifying the login settings for invoking the Content Repository Connector for IBM Content Manager service.

*Login Mode*

A `LoginMode` (`com.adobe.livecycle.connectorforibmcm.dsc.client.type.LoginMode`) value that represents how the operation will be authenticated with IBM Content Manager. Valid values are `Use Credentials From Process Context` and `Use User Credentials`.

**NOTE:** `Use Credentials From Process Context` works only when both IBM Content Manager and AEM Forms are synchronized to the same LDAP server.

If you provide a literal value, select one of the following options.

**Use Credentials From Process Context:**

Uses the AEM Forms User Management credential from the process context for the IBM Content Manager authentication. The User Name, Data Store, Password, and Connection String properties are not used.

**NOTE:** To invoke a process with Workbench that uses this operation, type the login name in the Username box using the format *`[User Name]`*`@`*`[Data Store]`*`:`*`[Connection String]`*. In this format, *`[User Name]`* identifies the user ID, *`[Data Store]`* is the name of the repository to connect to, and *`[Connection String]`* is the list of options to use to connect to the repository.

**Use User Credentials:**

Uses the User Name and Password properties to authenticate with IBM Content Manager. Ensure that the user name is valid for the repository that is specified in the Data Store property for this operation.

When this value is selected, provide values for the Data Store, User Name, Password, and Connection String properties.

*Data Store*

(Optional) A `string` value that specifies the name of the data store that the user is to connect to.

This field is used only when the Login Mode property is set to `Use User Credentials`.

If you provide a literal value, type the name of the data store.

*User Name*

(Optional) A `string` value that specifies the user name to use when this operation is authenticated with IBM Content Manager No default is provided.

This field is used only when the Login Mode property is set to `Use User Credentials`.

If you provide a literal value, type the user ID.

*Password*

(Optional) A `string` value that specifies the password associated with the user name that is specified in the User Name property. The password is used when this operation is authenticated with IBM Content Manager. No default is provided.

This field is used only when the Login Mode property is set to `Use User Credentials`.

If you provide a literal value, type the password.

*Connection String*

(Optional) A `string` value that specifies additional options to use for connecting to IBM Content Manager, such as `SCHEMA=`*[schemaName]*, where *[schemaName]* is a case-sensitive value that represents a schema.

If you provide a literal value, type the additional connection options.

**Item Update Settings properties**

Properties for the item to update in the repository.

*PID*

A `string` value that represents the unique identifier (PID) of the item being updated. The PID (Persistent Identifier) is an identifier used by the IBM Content Manager server to uniquely identify all items.

If you provide a literal value, type the value that represents the PID.

*MIME Type*

A `string` value that specifies the MIME type of the item being updated in the repository. If the item being updated is a folder, the value in this property is ignored.

If you provide a literal value, type the value that represents the MIME type.

*Item Content*

(Optional) A `document` value that contains the new content used to update an existing item in the repository. If a value is not provided, the content of the item in the repository is not updated.

If you provide a literal value, clicking the ellipsis button displays the Select Asset dialog box. (See AboutSelect Asset.)

*New Version*

A `boolean` value that sets whether to create a version of the item that is being updated.

If you provide a literal value, select the New Version check box to create a version when updating the item in the repository. Deselect the check box to update the existing version of the item in the repository. The default for the New Version check box is deselected.

**NOTE:** This property is mandatory for the operation to run correctly.

**Meta-Data properties**

Properties to update the specified item with.

*Item Type*

A `string` value that represents the item type of the item to update in the repository. The value specified in this property is used to populate the `map` value in the Attributes property so that you can assign values

from process variables to attributes for the item type. However, if the item type is different from the item type of the item that is configured in the PID property, this value is ignored.

If you provide a literal value, type the value that represents the item type.

**NOTE:** When no value is provided, no attributes are populated in the *`map`* value in the Attributes property.

*Attributes*

A `map` value for assigning process variable values to an attribute of an item type. The process variables that you map to attributes must be the same data type. For example, if an attribute is of type `string`, it must be mapped to a process variable of data type `string`. The name of an attribute is used as the key.

If you provide a literal value, the list of attributes in the table is based on the item you specified in the Item Type property. You can select a process variable from the Value column to assign to each attribute in the Name column.

**NOTE:** No value is provided for this property when no value exists in the Item Type property.

**Results properties**

Properties to store the results of the operation.

*Document Information*

The attributes and contents of the item that was updated. The data type is `DocInfo`.

*PID*

The location to store the value that represents the unique identifier (PID) of the item that was updated. The PID (Persistent Identifier) is an identifier used by the IBM Content Manager server to uniquely identify all items. The data type is `string`.

*Item Type*

The location to store the value that represents the item type of the item that was updated. The data type is `string`.

*Item Version*

The location to store the value that represents the version of the item that was updated or created due to the update. The data type is `string`.

*Item Content*

The location to store the value that represents the contents of the update item. The data type is `document`.

*Attributes*

The attribute names and values of the item type for the updated item. The data type is `map`.

### Exceptions

This operation can throw a RepositoryException exception.

## Content Repository Connector for IBM Content Manager exceptions

The Content Repository Connector for IBM Content Manager service provides the following exceptions for throwing exception events.

### RepositoryException

Thrown if an error occurs while performing an operation on the IBM Content Manager repository. An error can occur for the following reasons:

- A valid PID does not exist in the specified repository. The PID is an identifier used by the IBM Content Manager server to uniquely identify all items.
- A relation type does not exist to create a relationship between two items in the IBM Content Manager repository.
- A relation type that was specified is a reserved word.
- A data store could not be found.
- A MIME type was specified that is not supported.
- A specified user name is invalid.
- An invalid password for a valid user name was specified.

# 22.15. Content Repository Connector for IBM FileNet

Lets you create processes that interact with content that is stored in an IBM FileNet repository. (See IBM FileNet P8 documentation

.)

## Content Repository Connector for IBM FileNet service configuration

The following service configuration property can be set for this service. (See Editing service configurations.)

### Asset Link Object Default Path:

The default portion of the path in the IBM FileNet repository for storing the Asset Link object. The actual path consists of the default path and the location of the form template in the AEM Forms repository.

For example, if the default path is set to `/AEMforms/ConnectorforIBMFileNet/AssetLinkObjects`, and the form template is stored in folder `/Docbase/forms/`, the Asset Link object is stored at the following location:

```
/AEMforms/ConnectorforIBMFileNet/AssetLinkObjects/Docbase/forms/
```

## Create folders operation

Creates a folder in theIBM FileNet repository.

For information about the General and Route Evaluation property groups, see Commonoperation properties.

### Login Settings properties

Login settings for invoking the Content Repository Connector for IBM FileNet service.

*Login Mode*

A `com.adobe.livecycle.ibmfilenetcontentrepositoryconnector.client.type.impl.LoginSettings` value that represents how the operation will log in to the FileNet repository. Valid values are `INVOCATION_CONTEXT` (Use Credentials from process context), `USER_CREDENTIALS` (Use User Credentials), and `TOKEN`(Use FileNet Credentials Token).

**Use Credentials from process context:**

Uses the AEM Forms User Management credential from the process context for the FileNet authentication.

**Use User Credentials:**

Uses the User Name and Password properties to authenticate with FileNet. Ensure that the user name is valid for the FileNet P8 Content Engine that is specified in administration console.

**Use FileNet Credentials Token:**

Uses the FileNet Credentials Token property to authenticate with the FileNet Content Engine. This login token must be valid in the current FileNet session.

*User Name*

A `string` value that specifies the account name to use when connecting to the FileNet object store.

This field is used only when the Login Mode property is set to `USER_CREDENTIALS` (Use User Credentials) or `TOKEN` (Use FileNet Credentials Token).

*Password*

A `string` value that specifies the password that is associated with the account name that is specified in the User Name property to use when connecting to the FileNet object store.

This field is used only when the Login Mode property is set to `USER_CREDENTIALS` (Use User Credentials).

*FileNet Credentials Token*

A `string` value that represents the location of an existing FileNet credentials token.

This field is used only when the Login Mode property is set to `TOKEN` (Use FileNet Credentials Token).

### Object Store and Folder Creation Settings properties

Properties to use when creating a folder in the repository.

*Object Store*

A `string` value that represents the name of the IBM FileNet repository. You can choose from the list of available object stores from your current FileNet Content Engine defined in administration console. (See AEM Forms administration help

.) No default is provided.

*Folder Path*

A `string` value that indicates the folder path within the repository where you want to create the folder.

If you provide a literal value, type the path or click Browse to select the folder from the IBM FileNet repository.

When specifying the folder path, use the following rules:

- Do not include the object store name in the folder path.
- Use a forward slash (/) to separate folder names if you specify a folder location, as shown in this example:

  ```
  /[FolderPath]/[New Folder Name]
  ```

**NOTE:** Folders that are specified in the folder path that do not exist are created automatically.

### Result properties

Properties for the created folder.

*leafFolderGuid*

A `string` value that specifies the GUID of the created folder in the folder path.

### Exceptions

This operation can throw a RepositoryException exception.

## Create Relationship operation

Creates a relationship between two documents that are present in a IBM FileNet repository.

For information about the General and Route Evaluation property groups, see Commonoperation properties.

### Login Settings properties

Login settings for invoking the Content Repository Connector for IBM FileNet service.

*Login Mode*

A `com.adobe.livecycle.ibmfilenetcontentrepositoryconnector.client.type.impl.LoginSettings` value that represents how the operation will log in to the FileNet repository. Valid values are `INVOCATION_CONTEXT` (Use Credentials from process context), `USER_CREDENTIALS` (Use User Credentials), and `TOKEN`(Use FileNet Credentials Token).

**Use Credentials from process context:**

Uses the AEM Forms User Management credential from the process context for the FileNet authentication.

**Use User Credentials:**

Uses the User Name and Password properties to authenticate with FileNet. Ensure that the user name is valid for the FileNet P8 Content Engine that is specified in administration console.

**Use FileNet Credentials Token:**

Uses the FileNet Credentials Token property to authenticate with the FileNet Content Engine. This login token must be valid in the current FileNet session.

*User Name*

A `string` value that specifies the account name to use when connecting to the FileNet object store.

This field is used only when the Login Mode property is set to `USER_CREDENTIALS` (Use User Credentials) or `TOKEN` (Use FileNet Credentials Token).

*Password*

A `string` value that specifies the password that is associated with the account name that is specified in the User Name property to use when connecting to the FileNet object store.

This field is used only when the Login Mode property is set to `USER_CREDENTIALS` (Use User Credentials).

*FileNet Credentials Token*

A `string` value that represents the location of an existing FileNet credentials token.

This field is used only when the Login Mode property is set to `TOKEN` (Use FileNet Credentials Token).

**Relationship Creation Settings properties**

Specifies the properties for the relationship you want to establish.

*Object Store*

A `string` value that represents the name of the IBM FileNet repository. You can choose from the list of available object stores from your current FileNet Content Engine defined in administration console. (See AEM Forms administration help.)

*Relationship Type*

A `string` value that specifies the type of relationship to create between the head document and the tail document.

*Head Document GUID Or Path*

A `string` value that indicates either the head document GUID or the folder location within the object store where the head document exists.

If you provide a literal value, type the path or click Browse to select the folder from the IBM FileNet repository.

When typing the folder path, use the following rules:

- Do not include the object store name in the folder path.
- Use a forward slash(/) to separate folder names if you specify a folder location, as shown in this example:

  ```
  /[FolderPath]/[documentName]
  ```

*Tail Document GUID Or Path*

A `string` value that indicates either the tail document GUID number or the folder location within the object store where the tail document exists.

If you provide a literal value, type the path or click Browse to select the folder from the IBM FileNet repository.

When typing the folder path, use the following rules:

- Do not include the object store name in the folder path.
- Use a forward slash (/) to separate folder names if you specify a folder location, as shown in this example:

  ```
  /[FolderPath]/[documentName]
  ```

**Meta-Data properties**

Metadata properties for the documents.

*Property Mapping Table*

Map the document attributes to process variables in the given table. While creating or updating the document in the FileNet object store, the mapped document attributes are set on the document. If you leave an attribute value blank, the current value of that attribute in the repository is maintained. To clear the value of an attribute, map the attribute to "" (empty string).

**Exceptions properties**

This operation can throw a RepositoryException exception.

## Delete Content opreation

Deletes specified document or folder from the IBM FileNet repository. If a folder contains subfolders, they are also deleted.

For information about the General and Route Evaluation property groups, see Commonoperation properties.

### Login Settings properties

Login settings for invoking the Content Repository Connector for IBM FileNet service.

*Login Mode*

A `com.adobe.livecycle.ibmfilenetcontentrepositoryconnector.client.type.impl.LoginSettings` value that represents how the operation will log in to the FileNet repository. Valid values are `INVOCATION_CONTEXT` (Use Credentials from process context), `USER_CREDENTIALS` (Use User Credentials), and `TOKEN`(Use FileNet Credentials Token).

**Use Credentials from process context:**

Uses the AEM Forms User Management credential from the process context for the FileNet authentication.

**Use User Credentials:**

Uses the User Name and Password properties to authenticate with FileNet. Ensure that the user name is valid for the FileNet P8 Content Engine that is specified in administration console.

**Use FileNet Credentials Token:**

Uses the FileNet Credentials Token property to authenticate with the FileNet Content Engine. This login token must be valid in the current FileNet session.

*User Name*

A `string` value that specifies the account name to use when connecting to the FileNet object store.

This field is used only when the Login Mode property is set to `USER_CREDENTIALS` (Use User Credentials) or `TOKEN` (Use FileNet Credentials Token).

*Password*

A `string` value that specifies the password that is associated with the account name that is specified in the User Name property to use when connecting to the FileNet object store.

This field is used only when the Login Mode property is set to `USER_CREDENTIALS` (Use User Credentials).

*FileNet Credentials Token*

A `string` value that represents the location of an existing FileNet credentials token.

This field is used only when the Login Mode property is set to `TOKEN` (Use FileNet Credentials Token).

**Document/Folder Settings properties**

The repository properties for the document or folder to delete.

*Object Store*

A `string` value that represents the name of the IBM FileNet repository. You can choose from the list of available object stores from your current FileNet Content Engine defined in administration console. (See AEM Forms administration help.)

*Document GUID or Path*

A `string` value that indicates either the document GUID or the folder location within the object store of the document you want to delete.

If you provide a literal value, type the path or click Browse to select the folder from the IBM FileNet repository.

When typing the folder path, use the following rules:

- Do not include the object store name in the folder path.
- Use a forward slash (/) to separate folder names if you specify a folder location, as shown in this example:

  ```
  /[FolderPath]/[documentName]
  ```

**Exceptions**

This operation can throw a RepositoryException exception.

## Get Linked LC Assets Location operation

Retrieves the AEM Forms repository location of the form template for the specified form data object in the IBM FileNet repository.

For information about the General and Route Evaluation property groups, see Commonoperation properties.

**Login Settings properties**

Login settings for invoking the Content Repository Connector for IBM FileNet service.

*Login Mode*

A `com.adobe.livecycle.ibmfilenetcontentrepositoryconnector.client.type.impl.LoginSettings` value that represents how the operation will log in to the FileNet repository. Valid values are `INVOCATION_CONTEXT` (Use Credentials from process context), `USER_CREDENTIALS` (Use User Credentials), and `TOKEN`(Use FileNet Credentials Token).

**Use Credentials from process context:**

Uses the AEM Forms User Management credential from the process context for the FileNet authentication.

**Use User Credentials:**

Uses the User Name and Password properties to authenticate with FileNet. Ensure that the user name is valid for the FileNet P8 Content Engine that is specified in administration console.

**Use FileNet Credentials Token:**

Uses the FileNet Credentials Token property to authenticate with the FileNet Content Engine. This login token must be valid in the current FileNet session.

*User Name*

A `string` value that specifies the account name to use when connecting to the FileNet object store.

This field is used only when the Login Mode property is set to `USER_CREDENTIALS` (Use User Credentials) or `TOKEN` (Use FileNet Credentials Token).

*Password*

A `string` value that specifies the password that is associated with the account name that is specified in the User Name property to use when connecting to the FileNet object store.

This field is used only when the Login Mode property is set to `USER_CREDENTIALS` (Use User Credentials).

*FileNet Credentials Token*

A `string` value that represents the location of an existing FileNet credentials token.

This field is used only when the Login Mode property is set to `TOKEN` (Use FileNet Credentials Token).

**Object Store and Form Data Settings properties**

Repository properties for the ECM object.

*Object Store*

A `string` value that represents the name of the IBM FileNet repository. You can choose from the list of available object stores from your current FileNet Content Engine defined in administration console. (See AEM Forms administration help.)

*ECM Object Path*

A `string` value that indicates the location of the ECM object within the repository.

If you provide a literal value, type the path or click Browse to select the folder from the IBM FileNet repository.

When typing the folder path, use the following rules:

- Do not include the object store name in the folder path.
- Use a forward slash (/) to separate folder names if you specify a folder location, as shown in this example:

  `/[FolderPath]/[documentName]`

**Relationship Type Settings properties**

Properties of the relationship between the ECM object and the Assets Link object.

*Relationship Between Form Data and Asset Link Object*

(Optional) A `string` value that represents the relationship between the ECM object and the Asset Link object.

**Output properties**

Properties of the Asset Link object.

*outGetLinkedLCAssetssLocationResult*

A `IGetLinkedLCAssetsLocationResult` value that contains information about the form template path, form template folder path, and the form template name.

*Linked LC Asset's URL*

A `string` value that represents the full path to the form template in the AEM Forms repository.

*Linked LC Asset's Folder Path*

A `string` value that represents the path to the folder containing the form template in the AEM Forms repository.

*Linked LC Asset's Name*

A `string` value that represents the name of the form template in the AEM Forms repository.

**Exceptions**

This operation can throw a RepositoryException exception.

## Get Related operation

Retrieves a list of IDs for all documents that are related to the specified document.

For information about the General and Route Evaluation property groups, see Commonoperation properties.

**Login Settings properties**

Login settings for invoking the Content Repository Connector for IBM FileNet service.

*Login Mode*

A `com.adobe.livecycle.ibmfilenetcontentrepositoryconnector.client.type.impl.LoginSettings` value that represents how the operation will log in to the FileNet repository. Valid values are `INVOCATION_CONTEXT` (Use Credentials from process context), `USER_CREDENTIALS` (Use User Credentials), and `TOKEN`(Use FileNet Credentials Token).

**Use Credentials from process context:**

Uses the AEM Forms User Management credential from the process context for the FileNet authentication.

**Use User Credentials:**

Uses the User Name and Password properties to authenticate with FileNet. Ensure that the user name is valid for the FileNet P8 Content Engine that is specified in administration console.

**Use FileNet Credentials Token:**

Uses the FileNet Credentials Token property to authenticate with the FileNet Content Engine. This login token must be valid in the current FileNet session.

*User Name*

A `string` value that specifies the account name to use when connecting to the FileNet object store.

This field is used only when the Login Mode property is set to `USER_CREDENTIALS` (Use User Credentials) or `TOKEN` (Use FileNet Credentials Token).

*Password*

A `string` value that specifies the password that is associated with the account name that is specified in the User Name property to use when connecting to the FileNet object store.

This field is used only when the Login Mode property is set to `USER_CREDENTIALS` (Use User Credentials).

*FileNet Credentials Token*

A `string` value that represents the location of an existing FileNet credentials token.

This field is used only when the Login Mode property is set to `TOKEN` (Use FileNet Credentials Token).

**Relationship Information properties**

Specifies the properties for the relationship you want to establish.

*Object Store*

A `string` value that represents the name of the FileNet object store. You can choose from the list of available object stores from your current FileNet Content Engine defined in administration console. (See AEM Forms administration help.)

*Relationship Type*

A `string` value that specifies the type of relationship to create between the head document and the tail document.

*Document GUID Or Path*

A `string` value that indicates either the document GUID or the folder location within the object store of the document that you want to refer to.

If you provide a literal value, type the path or click Browse to select the folder from the IBM FileNet repository.

When typing the folder path, use the following rules:

- Do not include the object store name in the folder path.
- Use a forward slash (/) to separate folder names if you specify a folder location, as shown in this example:

  ```
  /[FolderPath]/[documentName]
  ```

*Is Head in Relationship?*

A `boolean` value that indicates whether to return all documents that participate as tails in the relationship (`True`) or all documents that participate as heads in the relationship (`False`).

**Result properties**

Specifies the property for the result of the Get Related operation.

*Get Related Result*

A `list` that contains the IDs of documents that is related to the specified document.

For information about retrieving values from a list, see Accessingdata in data collections.

**Exceptions**

This operation can throw a RepositoryException exception.

## Retrieve Content operation

Retrieves a document from a IBM FileNet repository.

For information about the General and Route Evaluation property groups, see Commonoperation properties.

**Login Settings properties**

Login settings for invoking the Content Repository Connector for IBM FileNet service.

*Login Mode*

A `com.adobe.livecycle.ibmfilenetcontentrepositoryconnector.client.type.impl.LoginSettings` value that represents how the operation will log in to the FileNet repository. Valid values are `INVOCATION_CONTEXT` (Use Credentials from process context), `USER_CREDENTIALS` (Use User Credentials), and `TOKEN`(Use FileNet Credentials Token).

**Use Credentials from process context:**

Uses the AEM Forms User Management credential from the process context for the FileNet authentication.

**Use User Credentials:**

Uses the User Name and Password properties to authenticate with FileNet. Ensure that the user name is valid for the FileNet P8 Content Engine that is specified in administration console.

**Use FileNet Credentials Token:**

Uses the FileNet Credentials Token property to authenticate with the FileNet Content Engine. This login token must be valid in the current FileNet session.

*User Name*

A `string` value that specifies the account name to use when connecting to the FileNet object store.

This field is used only when the Login Mode property is set to `USER_CREDENTIALS` (Use User Credentials) or `TOKEN` (Use FileNet Credentials Token).

*Password*

A `string` value that specifies the password that is associated with the account name that is specified in the User Name property to use when connecting to the FileNet object store.

This field is used only when the Login Mode property is set to `USER_CREDENTIALS` (Use User Credentials).

*FileNet Credentials Token*

A `string` value that represents the location of an existing FileNet credentials token.

This field is used only when the Login Mode property is set to `TOKEN` (Use FileNet Credentials Token).

**Document Settings properties**

Specifies the repository properties for the document to retrieve.

*Object Store*

A `string` value that represents the name of the IBM FileNet repository. You can choose from the list of available object stores from your current FileNet Content Engine defined in administration console. (See AEM Forms administration help.)

*Document GUID Or Path*

A `string` value that indicates either the document GUID or the folder location within the object store.

If you provide a literal value, type the path or click Browse to select the folder from the IBM FileNet repository.

When typing the folder path, use the following rules:

- Do not include the object store name in the folder path.
- Use a forward slash (/)to separate folder names if you specify a folder location, as shown in this example:

  ```
  /[FolderPath]/[documentName]
  ```

**Meta-Data properties**

Specifies document metadata properties for the retrieved document.

*Document Class Type*

(Optional) A `string` value that represents the FileNet object type to retrieve from the repository. The value specified in this property is used to populate the map value in the Property Mapping Table property. This action enables you to assign values from process variables to attributes for the document class type.

If you provide a literal value, type the value that represents the document class type.

**NOTE:** When no value is provided, no attributes are populated in the `map` value in the Property Mapping Table property.

*Property Mapping Table*

A `map` value for mapping attributes to process variable values. The attributes are available only when a value is specified in the Document Class Type property. The process variables that you map to attributes must be the same data type. For example, if an attribute is a string, it must be mapped to a process variable of data type string. The name of an attribute is used as the key. During document retrieval, the mapped process variables is set with the value of the corresponding attributes from the document.

If you provide a literal value, the list of attributes in the table is based on the item you specified in the Document Class Type property. You can select a process variable from the Value column to assign to each attribute in the Name column.

**NOTE:** No value is provided for this property when no value exists in the Item Type property.

**Results properties**

Specifies the properties for the document that is retrieved from the IBM FileNet repository.

*Retrieve Content Result*

An `object` value that stores the content properties that are specified in the Results property group.

*Document Contents*

A `document` value that contains the document content.

*Document ID*

A `string` value that indicates the ID for the document.

*Document Major Version*

A `string` value that specifies the major version number of the document.

*Document Minor Version*

A `string` value that specifies the minor version number of the document.

*Document Class Type*

A `string` value that indicates the FileNet object type for the document.

*Document Mime Type*

A `string` value that indicates the MIME type of the document.

*Document Content Type*

A `string` value that indicates the FileNet content type for the document.

*Name of the Document Creator*

A `string` value that indicates the user name of the original author of the document.

*Document Creation Date*

A `date` value that indicates when the original version of the document was created.

*Last Document Modifier*

A `string` value that specifies the user name of the last person to change the document.

*Last Document Modification Date*

A `date` value that indicates when the document was last changed.

*Property Name Value Map*

A `map` value that contains values of attributes that are specified during document retrieval. You can also map individual attributes to process variables in the Meta-Data property group.

For information about retrieving values from a map, see Accessingdata in data collections.

### Exceptions

This operation can throw a RepositoryException exception.

## Set Link to LC Assets operation

Creates an Asset Link Object (ALO) object that establishes a link between objects, files, or data in the AEM forms repository and assets in the IBM FileNet repository. The properties of the ALO object cannot be modified.

For information about the General and Route Evaluation property groups, see Commonoperation properties.

### Login Settings properties

Login settings for invoking the Content Repository Connector for IBM FileNet service.

*Login Mode*

A `com.adobe.livecycle.ibmfilenetcontentrepositoryconnector.client.type.impl.LoginSettings` value that represents how the operation will log in to the FileNet repository. Valid values are `INVOCATION_CONTEXT` (Use Credentials from process context), `USER_CREDENTIALS` (Use User Credentials), and `TOKEN`(Use FileNet Credentials Token).

**Use Credentials from process context:**

Uses the AEM Forms User Management credential from the process context for the FileNet authentication.

**Use User Credentials:**

Uses the User Name and Password properties to authenticate with FileNet. Ensure that the user name is valid for the FileNet P8 Content Engine that is specified in administration console.

**Use FileNet Credentials Token:**

Uses the FileNet Credentials Token property to authenticate with the FileNet Content Engine. This login token must be valid in the current FileNet session.

*User Name*

A `string` value that specifies the account name to use when connecting to the FileNet object store.

This field is used only when the Login Mode property is set to `USER_CREDENTIALS` (Use User Credentials) or `TOKEN` (Use FileNet Credentials Token).

*Password*

A `string` value that specifies the password that is associated with the account name that is specified in the User Name property to use when connecting to the FileNet object store.

This field is used only when the Login Mode property is set to `USER_CREDENTIALS` (Use User Credentials).

*FileNet Credentials Token*

A `string` value that represents the location of an existing FileNet credentials token.

This field is used only when the Login Mode property is set to `TOKEN` (Use FileNet Credentials Token).

**Object Store and ECM Object Settings properties**

Repository properties for the ECM object.

*Object Store*

A `string` value that represents the name of the IBM FileNet repository. You can choose from the list of available object stores from your current FileNet Content Engine defined in administration console. (See AEM Forms administration help.)

*ECM Object Complete Path*

(Optional) A `string` value that indicates the location of the ECM object within the repository.

If you provide a literal value, type the path or click Browse to select the folder from the IBM FileNet repository.

When typing the folder path, use the following rules:

- Do not include the object store name in the folder path.
- Use a forward slash (/) to separate folder names if you specify a folder location, as shown in this example:

  ```
  /[FolderPath]/[documentName]
  ```

**Form Template Settings properties**

Repository properties for the form template.

*Repository Path Where Form Template is Kept*

A `string` value that indicates the location of the form template in the AEM Forms repository.

If you provide a literal value, type the path or click Browse to select the folder from the AEM Forms repository.

When typing the folder path, use a forward slash (/) to separate folder names if you specify a folder location, as shown in this example:

```
/[FolderPath]/[documentName]
```

**Relationship Settings properties**

Properties related to the relationship between the ECM object and the object in the AEM Forms repository.

*Relationship Type*

(Optional) A `string` value that represents the type of relationship between the ECM object and the object in the AEM Forms repository. The value specified in this property is used to populate the `map` value in the Property Mapping Table property. This action enables you to assign values from process variables to attributes for the relationship type.

If you provide a literal value, type the value that represents the relationship type.

**NOTE:** When no value is provided, no attributes are populated in the map value in the Attributes property.

*Property Mapping Table*

(Optional) A `map` value for mapping attributes to process variable values. The attributes are available only when a value is specified in the Relationship Type property. The process variables that you map to attributes must be the same data type. For example, if an attribute is a `string`, it must be mapped to a process variable of data type `string`. The name of an attribute is used as the key.

If you provide a literal value, the list of attributes in the table is based on the item you specified in the Relationship Type property. You can select a process variable from the Value column to assign to each attribute in the Name column.

**NOTE:** No value is provided for this property when no value exists in the Relationship Type property.

### Exceptions

This operation can throw a RepositoryException exception.

## Store Content operation

Stores a new document or updates an existing document in a IBM FileNet repository. This operation supports storing multi-valued properties (See Meta-Data.)

For information about the General and Route Evaluation property groups, see Commonoperation properties.

### Login Settings properties

Login settings for invoking the Content Repository Connector for IBM FileNet service.

*Login Mode*

A `com.adobe.livecycle.ibmfilenetcontentrepositoryconnector.client.type.impl.LoginSettings` value that represents how the operation will log in to the FileNet repository. Valid values are `INVOCATION_CONTEXT` (Use Credentials from process context), `USER_CREDENTIALS` (Use User Credentials), and `TOKEN`(Use FileNet Credentials Token).

**Use Credentials from process context:**

Uses the AEM Forms User Management credential from the process context for the FileNet authentication.

**Use User Credentials:**

Uses the User Name and Password properties to authenticate with FileNet. Ensure that the user name is valid for the FileNet P8 Content Engine that is specified in administration console.

**Use FileNet Credentials Token:**

Uses the FileNet Credentials Token property to authenticate with the FileNet Content Engine. This login token must be valid in the current FileNet session.

*User Name*

A `string` value that specifies the account name to use when connecting to the FileNet object store.

This field is used only when the Login Mode property is set to `USER_CREDENTIALS` (Use User Credentials) or `TOKEN` (Use FileNet Credentials Token).

*Password*

A `string` value that specifies the password that is associated with the account name that is specified in the User Name property to use when connecting to the FileNet object store.

This field is used only when the Login Mode property is set to `USER_CREDENTIALS` (Use User Credentials).

*FileNet Credentials Token*

A `string` value that represents the location of an existing FileNet credentials token.

This field is used only when the Login Mode property is set to `TOKEN` (Use FileNet Credentials Token).

**Object Store and Folder Settings properties**

Properties that specify where a document should be stored.

*Object Store*

A `string` value that represents the name of the IBM FileNet repository. You can choose from the list of available object stores from your current FileNet Content Engine defined in administration console. (See AEM Forms administration help.)

*Folder Path*

A `string` value that indicates the folder location within the object store where you want to create the document.

If you provide a literal value, type the path or click Browse to select the folder from the IBM FileNet repository.

When typing the folder path, use the following rules:

- Do not include the object store name in the folder path.
- Use a forward slash (/)to separate folder names if you specify a folder location, as shown in this example:

`/[FolderPath]/[documentName]`

**Document Creation Settings properties**

Properties to use when creating a document to store in the repository.

*Document Class Type*

A `string` value that indicates the FileNet object type for the document.

**NOTE:** This property is used only if a document with the same name does not exist.

*Document Mime Type*

A `string` value that indicates the MIME type for the document.

*Title of the document*

A `string` value that indicates the document title. If a document with the same object name as in the path specified in the Folder Path property, this operation updates the document object. The update is done according to the rule that is defined in the Update Version Type property.

**Document Content Settings properties**

Properties that are related to the content of the document to store in the repository.

*Update Version Type*

(Optional) An `UpdateVersionType` value that represents how to increment the document version updates based on the document type. The following values are valid:

**Increment Major Version:**

Updates the version number with a major version increment.

**Increment Minor Version:**

Updates the version number with a minor version increment.

*Document Contents*

A `document` value that contains the document content.

If you provide a literal value, clicking the ellipsis button displays the Select Asset dialog box. (See AboutSelect Asset.)

**NOTE:** This property is mandatory for the operation to run correctly.

**Meta-Data properties**

Metadata properties for the stored document.

*Property Mapping Table*

(Optional) Map the document attributes to process variables in the given table. While creating or updating the document in the FileNet object store, the mapped document attributes are set on the document. If you leave an attribute value blank, the current value of that attribute in the repository is maintained. To clear the value of an attribute, map the attribute to "" (empty string).

To set the value of a multi-value property of a document, map the name of the property to a list value. For example, the document class type named CustomClassType and the multi-value property called Array Property have been created in FileNet. CustomClassType has Array Property as a property, and Array Property stores string values. A process variable named listVar is a list that contains string values. The following configurations are required to store the values of listVar in the multi-value property:

- Set the value of the Document Class Type property to CustomClassType.
- Set the following values in the Meta-Data table:
  - **Name:** Array Property
  - **Type:** string
  - **Value:** /process_data/listVar

**Results properties**

Properties for the created or updated document.

*Store Content Result*

An `object` that stores the content properties that are specified in the Results property group.

*Document ID*

A `string` value that indicates the ID for the document.

*Document Major Version*

A `string` value that specifies the major version number of the document.

*Document Minor Version*

A `string` value that specifies the minor version number of the document.

*Name of the Document Creator*

A `string` value that indicates the user name of the original author of the document.

*Document Creation Date*

A `date` value that indicates when the original version of the document was created.

*Last Document Modifier*

A `string` value that specifies the user name of the last person to change the document.

*Last Document Modification Date*

A `date` value that indicates when the document was last changed.

#### Exceptions

This operation can throw a RepositoryException exception.

### Content Repository Connector for IBM FileNet exceptions

The Content Repository Connector for IBM FileNet service provides the following exceptions for throwing exception events.

#### RepositoryException

Thrown if an error occurs while retrieving or storing a document in a FileNet object store. For example, if the value of an operation parameter is `NULL` or if FileNet returns an error.

## 22.16. Connector for Microsoft SharePoint

Lets you create processes that interact with content that is stored in a Microsoft SharePoint repository.

### Connector for Microsoft SharePoint service configuration

The following service configuration property can be set for this service. (See Editing service configurations.)

#### Username and password for a user with Impersonation permissions

Enter the username and password for the user account you created with Identity Impersonation permission. (See AEM Forms administration help

.)

### Create Folder operation

Creates a folder in the SharePoint document library. You can also create sub folders within an existing folder.

For information about the General and Route Evaluation property groups, see Commonoperation properties.

#### Login Settings properties

Login settings for invoking the Connector for Microsoft SharePoint service.

*Login Mode*

A `com.adobe.livecycle.crc.sharepoint.client.types.LoginSettings` value that represents how the operation will be authenticated with SharePoint. Valid values are `INVOCATION_CONTEXT` (Use Credentials from process context) and `USER_CREDENTIALS` (Use User Credentials).

If you provide a literal value, select one of the following options:

**Use Credentials from process context:**

Uses the AEM Forms User Management credential from the process context for SharePoint authentication.

**Use User Credentials:**

Uses the specified User Name and Password properties to authenticate with the SharePoint server.

*Hostname*

(Optional) A `string` value that specifies the hostname of the machine which hosts Microsoft SharePoint. You can also use the Hostname field to connect with any web application running on a port number other that 80 by specifying the hostname and port number in the format `[hostname]:[port number]`.

This field is only used when Login Mode is set to 'Use User Credentials'. No default is provided.

*User Name*

(Optional) A `string` value that specifies the user name to be used to connect with the SharePoint server. If the user account is of Windows local users or users configured using Forms Authentication on the SharePoint site, only the user name is required. However, domain users for Windows Authentication enabled sites must provide both the domain name and login name in the format `domainName\LoginName`.

This field is only used when the Login Mode is set to 'Use User Credentials'. No default is provided.

*Password*

(Optional) A `string` value that specifies the password to be used to connect with the SharePoint server. This field is only used when Login Mode is set to 'Use User Credentials'. No default is provided.

*Domain*

(Optional) A `string` value that specifies the domain in which the SharePoint server is present. This field is only used when Login Mode is set to 'Use User Credentials'.

**Site and Document Library Information properties**

Properties to specify the SharePoint site and document library.

*Site URL*

A `string` value that specifies the URL of the SharePoint site. Do not append a forward slash to the path. If you provide a literal value, type the URL of the SharePoint site or click Sites to browse existing SharePoint sites on the host machine. No default is provided.

*Document Library Name*

A `string` value that specifies the document library. If you provide a literal value, type the path or click Document Library to select the document library from the Microsoft SharePoint repository. Do not include the repository name in the folder path.

**Folder Creation Settings properties**

Properties to specify the folder to be created.

*Parent Folder URL*

(Optional) A `string` value that specifies the relative URL of the parent folder. To create a folder directly within a document library, leave this parameter blank. Do not append a forward slash to the path.

If you are providing a literal value, click Parent Folder to browse the existing folders at the selected document library.

*New Folder Name*

A `string` value that specifies the new folder name. Do not append or prefix the folder name with either “ /” or “ \”.

**Result properties**

A `string` value that specifies the relative URL of the folder created.

**Exceptions**

This operation can throw a Connector for Microsoft SharePoint exceptions exception.

## Create Document operation

Creates a document in the SharePoint repository. If the document exists, this operation returns the absolute URL of the existing document and a `false`flag.

For information about the General and Route Evaluation property groups, see Commonoperation properties.

**Login Settings properties**

Login settings for invoking the Connector for Microsoft SharePoint service.

*Login Mode*

A `com.adobe.livecycle.crc.sharepoint.client.types.LoginSettings` value that represents how the operation will be authenticated with SharePoint. Valid values are `INVOCATION_CONTEXT` (Use Credentials from process context) and `USER_CREDENTIALS` (Use User Credentials).

If you provide a literal value, select one of the following options:

**Use Credentials from process context:**

Uses the AEM Forms User Management credential from the process context for SharePoint authentication.

**Use User Credentials:**

Uses the specified User Name and Password properties to authenticate with the SharePoint server.

*Hostname*

(Optional) A `string` value that specifies the hostname of the machine which hosts Microsoft SharePoint. You can also use the Hostname field to connect with any web application running on a port number other that 80 by specifying the hostname and port number in the format `[hostname]:[port number]`.

This field is only used when Login Mode is set to 'Use User Credentials'. No default is provided.

*User Name*

(Optional) A `string` value that specifies the user name to be used to connect with the SharePoint server. If the user account is of Windows local users or users configured using Forms Authentication on the SharePoint site, only the user name is required. However, domain users for Windows Authentication enabled sites must provide both the domain name and login name in the format `domainName\LoginName`.

This field is only used when the Login Mode is set to 'Use User Credentials'. No default is provided.

*Password*

(Optional) A `string` value that specifies the password to be used to connect with the SharePoint server. This field is only used when Login Mode is set to 'Use User Credentials'. No default is provided.

*Domain*

(Optional) A `string` value that specifies the domain in which the SharePoint server is present. This field is only used when Login Mode is set to 'Use User Credentials'.

**Site and Document Library Information properties**

Properties to identify the SharePoint site and document library where the document is to be created.

*Site URL*

A `string` value that specifies the URL of the Microsoft SharePoint site. Do not append a forward slash to the path. If you provide a literal value, type the URL or click Sites to browse the existing SharePoint sites on the host machine. No default is provided.

*Document Library Name*

A `string` value that specifies the document library name. You can browse to view all existing document libraries at the selected site. If you provide a literal value, type the path or click Document Library to select the folder from the Microsoft SharePoint repository. Do not include the repository name in the folder path.

**Document Content Information properties**

Properties to specify the document to be added to the SharePoint repository.

*Document*

A `document` object that represents the document to be added to SharePoint repository. If you provide a literal value, type the path or click the ellipsis button to select the document from an application. Do not include the repository name in the folder path.

**Document Creation Settings properties**

Properties to identify the location and filename for the new document.

*Parent Folder URL*

(Optional) A `string` value that specifies the relative URL of the parent folder. If you want to create a document directly within a document library, leave this parameter as blank. Do not append a forward slash to the path. If you provide a literal value, type the path or click Parent Folder to browse the existing folders at the selected document library.

*New File Name*

A `string` value that specifies the new filename.

*Set all or none properties*

(Optional) A `boolean` value that specifies whether all properties are set in a single transaction for the new document. If selected, the operation is executed in a single transaction. As a result, either all or none of the properties of the newly created document are set. If deselected, a best attempt to set each property is made and all those properties which were correctly stated are set. Default: `true`.

**Properties**

(Optional) A `map` value that contains string values that specify the properties to be set for the created file.

### Result properties

A `CreateDocumentResultType` value containing the absolute URL of the document created, relative URL of the document created, and a `fileUploadSuccess` flag.

### Exceptions

This operation can throw a RepositoryException exception.

## Check In File operation

Checks in the specified file to the SharePoint repository.

If a document library allows updating both the minor and major versions, then selecting the Update Major Version option updates the major version of the document. If not, only the minor version of the document is updated.

If a document library does not allow any major/minor version support on documents, selecting the Update Major Version option does not affect the check-in operation.

For information about the General and Route Evaluation property groups, see Commonoperation properties.

### Login Settings properties

Login settings for invoking the Connector for Microsoft SharePoint service.

*Login Mode*

A `com.adobe.livecycle.crc.sharepoint.client.types.LoginSettings` value that represents how the operation will be authenticated with SharePoint. Valid values are `INVOCATION_CONTEXT` (Use Credentials from process context) and `USER_CREDENTIALS` (Use User Credentials).

If you provide a literal value, select one of the following options:

**Use Credentials from process context:**

Uses the AEM Forms User Management credential from the process context for SharePoint authentication.

**Use User Credentials:**

Uses the specified User Name and Password properties to authenticate with the SharePoint server.

*Hostname*

(Optional) A `string` value that specifies the hostname of the machine which hosts Microsoft SharePoint. You can also use the Hostname field to connect with any web application running on a port number other that 80 by specifying the hostname and port number in the format `[hostname]:[port number]`.

This field is only used when Login Mode is set to 'Use User Credentials'. No default is provided.

*User Name*

(Optional) A `string` value that specifies the user name to be used to connect with the SharePoint server. If the user account is of Windows local users or users configured using Forms Authentication on the SharePoint site, only the user name is required. However, domain users for Windows Authentication enabled sites must provide both the domain name and login name in the format `domainName\LoginName`.

This field is only used when the Login Mode is set to 'Use User Credentials'. No default is provided.

*Password*

(Optional) A `string` value that specifies the password to be used to connect with the SharePoint server. This field is only used when Login Mode is set to 'Use User Credentials'. No default is provided.

*Domain*

(Optional) A `string` value that specifies the domain in which the SharePoint server is present. This field is only used when Login Mode is set to 'Use User Credentials'.

**Site and Document Library Information properties**

Properties to specify the SharePoint site and document library.

*Site URL*

A `string` value that specifies the URL of the SharePoint site. Do not append a forward slash to the path. If you provide a literal value, type the URL of the SharePoint site or click Sites to browse existing SharePoint sites on the host machine. No default is provided.

*Document Library Name*

A `string` value that specifies the document library. If you provide a literal value, type the path or click Document Library to select the document library from the Microsoft SharePoint repository. Do not include the repository name in the folder path.

**File Information properties**

Properties to specify the file to check in.

*Select file to check in*

A `string` value that specifies the relative URL of the file to check in. Do not append a forward slash to the path. If you provide a literal value, type the path or click Select File to view existing files and folders at the selected document library.

**Version Update Settings properties**

Specifies how to update the version of the document in the repository.

*Update Major Version*

(Optional) A `boolean` value that specifies whether to update the major version for the document. This setting must match the setting at your SharePoint server's document library for all documents that are being checked in. If these versions do not match, the document is not checked in. Default value is `false`.

*Comment*

(Optional) A `string` value that contains the description that you can enter when checking the file in.

**Result properties**

A `boolean` value that indicates whether the check-in operation is successful. A `true` indicates that check-in is successful; `false` if the operation failed.

**Exceptions**

This operation can throw a RepositoryException exception.

## Check Out File operation

Checks out the file specified by the file path from the SharePoint server. If the file is already checked out by the same user, this method returns `false`; otherwise it checks out the file and returns `true`.

For information about the General and Route Evaluation property groups, see Commonoperation properties.

**Login Settings properties**

Login settings for invoking the Connector for Microsoft SharePoint service.

*Login Mode*

A `com.adobe.livecycle.crc.sharepoint.client.types.LoginSettings` value that represents how the operation will be authenticated with SharePoint. Valid values are `INVOCATION_CONTEXT` (Use Credentials from process context) and `USER_CREDENTIALS` (Use User Credentials).

If you provide a literal value, select one of the following options:

**Use Credentials from process context:**

Uses the AEM Forms User Management credential from the process context for SharePoint authentication.

**Use User Credentials:**

Uses the specified User Name and Password properties to authenticate with the SharePoint server.

*Hostname*

(Optional) A `string` value that specifies the hostname of the machine which hosts Microsoft SharePoint. You can also use the Hostname field to connect with any web application running on a port number other that 80 by specifying the hostname and port number in the format `[hostname]:[port number]`.

This field is only used when Login Mode is set to 'Use User Credentials'. No default is provided.

*User Name*

(Optional) A `string` value that specifies the user name to be used to connect with the SharePoint server. If the user account is of Windows local users or users configured using Forms Authentication on the SharePoint site, only the user name is required. However, domain users for Windows Authentication enabled sites must provide both the domain name and login name in the format `domainName\LoginName`.

This field is only used when the Login Mode is set to 'Use User Credentials'. No default is provided.

*Password*

(Optional) A `string` value that specifies the password to be used to connect with the SharePoint server. This field is only used when Login Mode is set to 'Use User Credentials'. No default is provided.

*Domain*

(Optional) A `string` value that specifies the domain in which the SharePoint server is present. This field is only used when Login Mode is set to 'Use User Credentials'.

**Site and Document Library Information properties**

Properties to specify the SharePoint site and document library.

*Site URL*

A `string` value that specifies the URL of the SharePoint site. Do not append a forward slash to the path. If you provide a literal value, type the URL of the SharePoint site or click Sites to browse existing SharePoint sites on the host machine. No default is provided.

*Document Library Name*

A `string` value that specifies the document library. If you provide a literal value, type the path or click Document Library to select the document library from the Microsoft SharePoint repository. Do not include the repository name in the folder path.

**File Information properties**

Properties to specify the file to be checked out.

*Select File to Check Out*

A `string` value that specifies the relative URL of the file to be checked out. Do not append a forward slash to the path. If you provide a literal value, type the path or click Select File to view existing files and folders at the selected document library.

If the file does not exist, an exception is thrown.

**Result properties**

A `boolean` value that indicates whether the check out operation was successful. A `true` indicates that the file was checked out successfully; a `false` if check out failed.

**Exceptions**

This operation can throw a RepositoryException exception.

## Cancel File Check Out operation

Cancels the checkout of the file identified by the file path. If the file is not already checked out, this operation returns a `false` and does nothing. Otherwise, this operation cancels the check-out and returns `true`.

For information about the General and Route Evaluation property groups, see Commonoperation properties.

**Login Settings properties**

Login settings for invoking the Connector for Microsoft SharePoint service.

*Login Mode*

A `com.adobe.livecycle.crc.sharepoint.client.types.LoginSettings` value that represents how the operation will be authenticated with SharePoint. Valid values are `INVOCATION_CONTEXT` (Use Credentials from process context) and `USER_CREDENTIALS` (Use User Credentials).

If you provide a literal value, select one of the following options:

**Use Credentials from process context:**

Uses the AEM Forms User Management credential from the process context for SharePoint authentication.

**Use User Credentials:**

Uses the specified User Name and Password properties to authenticate with the SharePoint server.

*Hostname*

(Optional) A `string` value that specifies the hostname of the machine which hosts Microsoft SharePoint. You can also use the Hostname field to connect with any web application running on a port

number other that 80 by specifying the hostname and port number in the format `[hostname]:[port number]`.

This field is only used when Login Mode is set to 'Use User Credentials'. No default is provided.

*User Name*

(Optional) A `string` value that specifies the user name to be used to connect with the SharePoint server. If the user account is of Windows local users or users configured using Forms Authentication on the SharePoint site, only the user name is required. However, domain users for Windows Authentication enabled sites must provide both the domain name and login name in the format `domainName\LoginName`.

This field is only used when the Login Mode is set to 'Use User Credentials'. No default is provided.

*Password*

(Optional) A `string` value that specifies the password to be used to connect with the SharePoint server. This field is only used when Login Mode is set to 'Use User Credentials'. No default is provided.

*Domain*

(Optional) A `string` value that specifies the domain in which the SharePoint server is present. This field is only used when Login Mode is set to 'Use User Credentials'.

**Site and Document Library Information properties properties**

Properties to specify the SharePoint site and document library.

*Site URL*

A `string` value that specifies the URL of the SharePoint site. Do not append a forward slash to the path. If you provide a literal value, type the URL of the SharePoint site or click Sites to browse existing SharePoint sites on the host machine. No default is provided.

*Document Library Name*

A `string` value that specifies the document library. If you provide a literal value, type the path or click Document Library to select the document library from the Microsoft SharePoint repository. Do not include the repository name in the folder path.

**File Location Information properties**

Properties to specify the file on which the check out operation is to be canceled.

*Select File*

A `string` value that specifies the relative URL of the file on which the check-out is to be canceled. Do not append a forward slash to the path. If you provide a literal value, type the path or click Select File to view all existing files at the selected document library. If the file does not exist, an exception is thrown.

### Result properties

A `boolean` value that indicates whether canceling the check out operation is successful. A `true` indicates success; a `false`if the check out failed.

### Exceptions

This operation can throw a RepositoryException exception.

## Search operation

Searches the SharePoint site for the specified string.

For information about the General and Route Evaluation property groups, see Commonoperation properties.

### Login Settings properties

Login settings for invoking the Connector for Microsoft SharePoint service.

*Login Mode*

A `com.adobe.livecycle.crc.sharepoint.client.types.LoginSettings` value that represents how the operation will be authenticated with SharePoint. Valid values are `INVOCATION_CONTEXT` (Use Credentials from process context) and `USER_CREDENTIALS` (Use User Credentials).

If you provide a literal value, select one of the following options:

**Use Credentials from process context:**

Uses the AEM Forms User Management credential from the process context for SharePoint authentication.

**Use User Credentials:**

Uses the specified User Name and Password properties to authenticate with the SharePoint server.

*Hostname*

(Optional) A `string` value that specifies the hostname of the machine which hosts Microsoft SharePoint. You can also use the Hostname field to connect with any web application running on a port number other that 80 by specifying the hostname and port number in the format `[hostname]:[port number]`.

This field is only used when Login Mode is set to 'Use User Credentials'. No default is provided.

*User Name*

(Optional) A `string` value that specifies the user name to be used to connect with the SharePoint server. If the user account is of Windows local users or users configured using Forms Authentication on the SharePoint site, only the user name is required. However, domain users for Windows Authentication

enabled sites must provide both the domain name and login name in the format `domainName\LoginName`.

This field is only used when the Login Mode is set to 'Use User Credentials'. No default is provided.

*Password*

(Optional) A `string` value that specifies the password to be used to connect with the SharePoint server. This field is only used when Login Mode is set to 'Use User Credentials'. No default is provided.

*Domain*

(Optional) A `string` value that specifies the domain in which the SharePoint server is present. This field is only used when Login Mode is set to 'Use User Credentials'.

**Context Scope Information properties**

Properties to identify the SharePoint site and the document library.

*Site URL*

A `string` value that specifies the URL of the SharePoint site. Do not append a forward slash to the path. Click Sites to browse the existing SharePoint sites on the host machine. No default is provided.

*Document Library Name*

(Optional) A `string` value that specifies the Document library Name. You can browse to view all existing document libraries at the selected site. If you provide a literal value, type the path or click Document Library to select the folder from the Microsoft SharePoint repository. Do not include the repository name in the folder path.

*Parent Folder*

(Optional) A `string` value that specifies the parent folder to search. Do not append a forward slash to the path. If you provide a literal value, type the path or click Parent Folder to view existing folders at the selected document library.

**Custom Scope Information properties properties**

Properties that specify the scope of the search.

*Custom Scope*

A `string` value that specifies a custom scope, such as search all sites.

**Query Creation Settings properties**

Properties to create the search string.

*Keyword*

A `string` value that specifies the keyword to search for.

*Maximum Results to Return*

(Optional) An `int` value that specifies the maximum number of results to return.

**Result properties**

A `SearchQueryResultType` value that contains the search result including the list of documents searched, number of search results returned, total search count, and the search status.

Each search document represents one document in SharePoint site with the following information about it: author, relative URL of the document, absolute URL path, and title. The absolute URL path is the absolute HTTP location of the document and can be used to retrieve the document.

If the search count is equal to the number of documents available, it indicates that the search operation has returned all the results. The parameter 'total available' indicates the total available results for the search query at SharePoint site. You can increase the value of maximum results if the search count is less than the total available.

**Exceptions**

This operation can throw a RepositoryException exception.

## Delete operation

Deletes the specified file or folder.

For information about the General and Route Evaluation property groups, see Commonoperation properties.

**Login Settings properties**

Login settings for invoking the Connector for Microsoft SharePoint service.

*Login Mode*

A `com.adobe.livecycle.crc.sharepoint.client.types.LoginSettings` value that represents how the operation will be authenticated with SharePoint. Valid values are `INVOCATION_CONTEXT` (Use Credentials from process context) and `USER_CREDENTIALS` (Use User Credentials).

If you provide a literal value, select one of the following options:

**Use Credentials from process context:**

Uses the AEM Forms User Management credential from the process context for SharePoint authentication.

**Use User Credentials:**

Uses the specified User Name and Password properties to authenticate with the SharePoint server.

*Hostname*

(Optional) A `string` value that specifies the hostname of the machine which hosts Microsoft SharePoint. You can also use the Hostname field to connect with any web application running on a port number other that 80 by specifying the hostname and port number in the format `[hostname]:[port number]`.

This field is only used when Login Mode is set to 'Use User Credentials'. No default is provided.

*User Name*

(Optional) A `string` value that specifies the user name to be used to connect with the SharePoint server. If the user account is of Windows local users or users configured using Forms Authentication on the SharePoint site, only the user name is required. However, domain users for Windows Authentication enabled sites must provide both the domain name and login name in the format `domainName\LoginName`.

This field is only used when the Login Mode is set to 'Use User Credentials'. No default is provided.

*Password*

(Optional) A `string` value that specifies the password to be used to connect with the SharePoint server. This field is only used when Login Mode is set to 'Use User Credentials'. No default is provided.

*Domain*

(Optional) A `string` value that specifies the domain in which the SharePoint server is present. This field is only used when Login Mode is set to 'Use User Credentials'.

**Site and Document Library Information properties**

Properties to specify the SharePoint site and document library.

*Site URL*

A `string` value that specifies the URL of the SharePoint site. Do not append a forward slash to the path. If you provide a literal value, type the URL of the SharePoint site or click Sites to browse existing SharePoint sites on the host machine. No default is provided.

*Document Library Name*

A `string` value that specifies the document library. If you provide a literal value, type the path or click Document Library to select the document library from the Microsoft SharePoint repository. Do not include the repository name in the folder path.

**File/Folder Location Settings properties**

Properties to specify the file or folder to delete.

*File/Folder to Delete*

A `string` value that specifies the relative URL of the file or folder to delete. Do not append a forward slash to the path. Click Select File to view all existing files and folders at the selected document library. If the file or folder does not exist, no exception is thrown.

**Exceptions**

This operation can throw a RepositoryException exception.

## Get Properties operation

Get properties of a document as a map value containing name-value pairs of document properties. Document content is not considered as a document's property.

For information about the General and Route Evaluation property groups, see Commonoperation properties.

**Login Settings properties**

Login settings for invoking the Connector for Microsoft SharePoint service.

*Login Mode*

A `com.adobe.livecycle.crc.sharepoint.client.types.LoginSettings` value that represents how the operation will be authenticated with SharePoint. Valid values are `INVOCATION_CONTEXT` (Use Credentials from process context) and `USER_CREDENTIALS` (Use User Credentials).

If you provide a literal value, select one of the following options:

**Use Credentials from process context:**

Uses the AEM Forms User Management credential from the process context for SharePoint authentication.

**Use User Credentials:**

Uses the specified User Name and Password properties to authenticate with the SharePoint server.

*Hostname*

(Optional) A `string` value that specifies the hostname of the machine which hosts Microsoft SharePoint. You can also use the Hostname field to connect with any web application running on a port number other that 80 by specifying the hostname and port number in the format `[hostname]:[port number]`.

This field is only used when Login Mode is set to 'Use User Credentials'. No default is provided.

*User Name*

(Optional) A `string` value that specifies the user name to be used to connect with the SharePoint server. If the user account is of Windows local users or users configured using Forms Authentication on the SharePoint site, only the user name is required. However, domain users for Windows Authentication enabled sites must provide both the domain name and login name in the format `domainName\LoginName`.

This field is only used when the Login Mode is set to 'Use User Credentials'. No default is provided.

*Password*

(Optional) A `string` value that specifies the password to be used to connect with the SharePoint server. This field is only used when Login Mode is set to 'Use User Credentials'. No default is provided.

*Domain*

(Optional) A `string` value that specifies the domain in which the SharePoint server is present. This field is only used when Login Mode is set to 'Use User Credentials'.

**Site and Document Library Information properties**

Properties to specify the SharePoint site and document library.

*Site URL*

A `string` value that specifies the URL of the SharePoint site. Do not append a forward slash to the path. If you provide a literal value, type the URL of the SharePoint site or click Sites to browse existing SharePoint sites on the host machine. No default is provided.

*Document Library Name*

A `string` value that specifies the document library. If you provide a literal value, type the path or click Document Library to select the document library from the Microsoft SharePoint repository. Do not include the repository name in the folder path.

**File Location Details properties**

Properties to specify the file for which the properties are to be updated.

*Select File*

A `string` value that specifies the file for which the properties are to be updated. Do not append a forward slash to the path. You can browse to view all existing files/folders at the selected document library.

**Result properties**

A `map` value that contains name—value pairs.

### Exceptions

This operation can throw a RepositoryException exception.

## Retrieve Document Content operation

Retrieves the contents of the specified document as a `document` object. No document properties are fetched in the process. To fetch document properties, use the Get Properties operation instead. This operation automatically determines mime-type of the document from the document extension. However, if the extension is missing, the default mime-type of the document is set to `application\octet-stream`.

For information about the General and Route Evaluation property groups, see Commonoperation properties.

### Login Settings properties

Login settings for invoking the Connector for Microsoft SharePoint service.

*Login Mode*

A `com.adobe.livecycle.crc.sharepoint.client.types.LoginSettings` value that represents how the operation will be authenticated with SharePoint. Valid values are `INVOCATION_CONTEXT` (Use Credentials from process context) and `USER_CREDENTIALS` (Use User Credentials).

If you provide a literal value, select one of the following options:

**Use Credentials from process context:**

Uses the AEM Forms User Management credential from the process context for SharePoint authentication.

**Use User Credentials:**

Uses the specified User Name and Password properties to authenticate with the SharePoint server.

*Hostname*

(Optional) A `string` value that specifies the hostname of the machine which hosts Microsoft SharePoint. You can also use the Hostname field to connect with any web application running on a port number other that 80 by specifying the hostname and port number in the format `[hostname]:[port number]`.

This field is only used when Login Mode is set to 'Use User Credentials'. No default is provided.

*User Name*

(Optional) A `string` value that specifies the user name to be used to connect with the SharePoint server. If the user account is of Windows local users or users configured using Forms Authentication on the SharePoint site, only the user name is required. However, domain users for Windows Authentication

enabled sites must provide both the domain name and login name in the format `domainName\LoginName`.

This field is only used when the Login Mode is set to 'Use User Credentials'. No default is provided.

*Password*

(Optional) A `string` value that specifies the password to be used to connect with the SharePoint server. This field is only used when Login Mode is set to 'Use User Credentials'. No default is provided.

*Domain*

(Optional) A `string` value that specifies the domain in which the SharePoint server is present. This field is only used when Login Mode is set to 'Use User Credentials'.

**Site and Document Library Information properties properties**

Properties to specify the SharePoint site and document library.

*Site URL*

A `string` value that specifies the URL of the SharePoint site. Do not append a forward slash to the path. If you provide a literal value, type the URL of the SharePoint site or click Sites to browse existing SharePoint sites on the host machine. No default is provided.

*Document Library Name*

A `string` value that specifies the document library. If you provide a literal value, type the path or click Document Library to select the document library from the Microsoft SharePoint repository. Do not include the repository name in the folder path.

**Document Identification properties**

Properties to specify the document of which the content is to be retrieved.

*Select File*

A `string` value that specifies the file from which the content is to be retrieved. Do not append a forward slash to the path. You can browse to view all existing files/folders at the selected document library. If the file does not exist, an exception is thrown.

**Result properties**

A `document` object that represents the contents of the specified document.

**Exceptions**

This operation can throw a RepositoryException exception.

## Set Document Content operation

Sets the contents of a document. This operation does not set the document properties of the document.

To modify the properties of a document, use the Update Properties operation. If the SharePoint document library requires that you check out the document before updating the document content, this operation throws an exception if the target document is not already checked out.

For information about the General and Route Evaluation property groups, see Commonoperation properties.

### Login Settings properties

Login settings for invoking the Connector for Microsoft SharePoint service.

*Login Mode*

A `com.adobe.livecycle.crc.sharepoint.client.types.LoginSettings` value that represents how the operation will be authenticated with SharePoint. Valid values are `INVOCATION_CONTEXT` (Use Credentials from process context) and `USER_CREDENTIALS` (Use User Credentials).

If you provide a literal value, select one of the following options:

**Use Credentials from process context:**

Uses the AEM Forms User Management credential from the process context for SharePoint authentication.

**Use User Credentials:**

Uses the specified User Name and Password properties to authenticate with the SharePoint server.

*Hostname*

(Optional) A `string` value that specifies the hostname of the machine which hosts Microsoft SharePoint. You can also use the Hostname field to connect with any web application running on a port number other that 80 by specifying the hostname and port number in the format `[hostname]:[port number]`.

This field is only used when Login Mode is set to 'Use User Credentials'. No default is provided.

*User Name*

(Optional) A `string` value that specifies the user name to be used to connect with the SharePoint server. If the user account is of Windows local users or users configured using Forms Authentication on the SharePoint site, only the user name is required. However, domain users for Windows Authentication enabled sites must provide both the domain name and login name in the format `domainName\LoginName`.

This field is only used when the Login Mode is set to 'Use User Credentials'. No default is provided.

*Password*

(Optional) A `string` value that specifies the password to be used to connect with the SharePoint server. This field is only used when Login Mode is set to 'Use User Credentials'. No default is provided.

*Domain*

(Optional) A `string` value that specifies the domain in which the SharePoint server is present. This field is only used when Login Mode is set to 'Use User Credentials'.

**Site and Document Library Information properties**

Properties to specify the SharePoint site and document library.

*Site URL*

A `string` value that specifies the URL of the SharePoint site. Do not append a forward slash to the path. If you provide a literal value, type the URL of the SharePoint site or click Sites to browse existing SharePoint sites on the host machine. No default is provided.

*Document Library Name*

A `string` value that specifies the document library. If you provide a literal value, type the path or click Document Library to select the document library from the Microsoft SharePoint repository. Do not include the repository name in the folder path.

**Document Identification properties**

*Select File*

A `string` value that specifies the document. Do not append a forward slash to the path. If you provide a literal value, type the path or click Select File to view existing files/folders at the selected document library. If the file does not exist, an exception is thrown.

**Document Content Information properties**

A `document` object representing the new content of the uploaded file.

**Result**

A `string` value that specifies the absolute URL of the modified document.

**Exceptions**

This operation can throw a RepositoryException exception.

## Update Properties operation

This operation updates the properties of the specified document. Document content is not considered as a document's property. To update the document content, use the Set Document Content operation instead.

For information about the General and Route Evaluation property groups, see Commonoperation properties.

### Login Settings properties

Login settings for invoking the Connector for Microsoft SharePoint service.

*Login Mode*

A `com.adobe.livecycle.crc.sharepoint.client.types.LoginSettings` value that represents how the operation will be authenticated with SharePoint. Valid values are `INVOCATION_CONTEXT` (Use Credentials from process context) and `USER_CREDENTIALS` (Use User Credentials).

If you provide a literal value, select one of the following options:

**Use Credentials from process context:**

Uses the AEM Forms User Management credential from the process context for SharePoint authentication.

**Use User Credentials:**

Uses the specified User Name and Password properties to authenticate with the SharePoint server.

*Hostname*

(Optional) A `string` value that specifies the hostname of the machine which hosts Microsoft SharePoint. You can also use the Hostname field to connect with any web application running on a port number other that 80 by specifying the hostname and port number in the format `[hostname]:[port number]`.

This field is only used when Login Mode is set to 'Use User Credentials'. No default is provided.

*User Name*

(Optional) A `string` value that specifies the user name to be used to connect with the SharePoint server. If the user account is of Windows local users or users configured using Forms Authentication on the SharePoint site, only the user name is required. However, domain users for Windows Authentication enabled sites must provide both the domain name and login name in the format `domainName\LoginName`.

This field is only used when the Login Mode is set to 'Use User Credentials'. No default is provided.

*Password*

(Optional) A `string` value that specifies the password to be used to connect with the SharePoint server. This field is only used when Login Mode is set to 'Use User Credentials'. No default is provided.

*Domain*

(Optional) A `string` value that specifies the domain in which the SharePoint server is present. This field is only used when Login Mode is set to 'Use User Credentials'.

**Site and Document Library Information**

Properties to specify the SharePoint site and document library.

*Site URL*

A `string` value that specifies the URL of the SharePoint site. Do not append a forward slash to the path. If you provide a literal value, type the URL of the SharePoint site or click Sites to browse existing SharePoint sites on the host machine. No default is provided.

*Document Library Name*

A `string` value that specifies the document library. If you provide a literal value, type the path or click Document Library to select the document library from the Microsoft SharePoint repository. Do not include the repository name in the folder path.

**Document Identification properties**

Properties to specify the document for which the properties are to be updated.

*File URL*

A `string` value that specifies the file for which the properties are to be updated. Do not append a forward slash to the path. If you provide a literal value, click Select File to view existing files/folders at the selected document library. If the file does not exist, an exception is thrown.

**Document Properties properties**

Properties to apply to the specified document.

*Properties*

A `map` value that contains the document properties (name—value pairs) to apply to the specified document.

*Use Same Transaction*

A `boolean` value that specifies whether all the document properties are updated in a single transaction. As a result, either all or none of the properties are set. If deselected, a best attempt to set each property is made and all those properties which were correctly stated is set. Default: `true`. If you provide a literal value, select or deselect the Use Same Transaction option.

#### Exceptions

This operation can throw a RepositoryException exception.

### Connector for Microsoft SharePoint exception

The Connector for Microsoft SharePoint service provides the following exception events.

#### RepositoryException

This exception is thrown if an error occurs while retrieving or storing a document in a Microsoft SharePoint repository (for example, if the value of an operation parameter is `NULL` or if SharePoint returns an error).

## 22.17. Convert PDF

Enables the automatic conversion of PDF documents to PostScript or image files.

For information about using the Convert PDF service, see AEM Forms Services

.

### toImage operation

Converts a PDF document to an image file. Supported image formats are JPEG, JPEG2000, PNG, and TIFF.

The following information applies to conversions to TIFF images:

- A multi-page TIFF file is generated.
- Some annotations are not included in TIFF images. Annotations that require Acrobat to generate their appearance are not included.

For information about the General and Route Evaluation property groups, see Commonoperation properties.

#### Input properties

Properties to specify the document to be converted and the conversion options.

*PDF Document*

The `document` value that represents the PDF document to be converted.

If you provide a literal value, clicking the ellipsis button displays the Select Asset dialog box. (See AboutSelect Asset.)

*Convert Image Options*

A `ToImageOptionsSpec` value that contains various preferences used to convert the PDF document to an image file. With the exception of the conversion format, which must be specified, default values for preferences are used if no value is specified.

If you provide a literal value, the availability of the following options depend on the value chosen for the Image Conversion Format option.

**Image Conversion Format:**

The format to convert an image to. The default is `JPEG`. The following values are valid.

- **JPEG:** The Joint Photographics Experts Group is a lossy image compression format.
- **JPEG2000:** The Joint Photographics Experts Group 2000 is a wavelet-based image compression format to replace the JPEG format.
- **PNG:** The Portable Network Graphics is a lossless image compression format.
- **TIFF:** The Tagged Image File Format is generally a lossless image compression format.

**Grayscale Compression:**

The gray scale compression to use. This option is available when the Image Conversion Format option is a value of JPEG, JPEG2000, or TIFF. No default is provided. The following values are valid.

- **Minimum:** The minimum level of gray scale compression.
- **Low:** A low level of gray scale compression.
- **Medium:** A medium level of gray scale compression.
- **High:** A high level of gray scale compression.
- **Maximum:** The maximum level of gray scale compression.

**Color Compression:**

The color compression to use. No default is provided. This option is available when the Image Conversion Format option is a value of JPEG, JPEG2000, or TIFF.

- **Minimum:** The minimum level of color compression.
- **Low:** A low level of color compression.
- **Medium:** A medium level of color compression.
- **High:** A high level of color compression.
- **Maximum:** The maximum level of color compression.

**JPEG Format:**

The JPEG format to use. The default is `Baseline Standard`. This option is available when the Image Conversion Format option is a value of JPEG.The following values are valid.

- **Baseline Standard:** Displays the image when it is fully downloaded. This JPEG format is recognizable to most web browsers.

- **Baseline Optimized:** Optimizes color quality of the image and produces smaller file sizes but is not supported by all web browsers.
- **3 scans:** Downloads the image first as a low-resolution image, with incremental quality improvements as downloading continues in three scans.
- **4 scans:** Downloads the image first as a low-resolution image, with incremental quality improvements as downloading continues in four scans.
- **5 scans:** Downloads the image first as a low-resolution image, with incremental quality improvements as downloading continues in five scans.

**RGB Policy:**

The type of color management typical for display on screens to be applied to the output file and whether to embed an ICC profile. The default is `Embed Profile`. The following values are valid.

- **Embed Profile:** Embed an ICC profile.
- **Off:** Do not embed an ICC profile.

**CMYK Policy:**

The type of color management typical for print to be applied to the output file and whether to embed an ICC profile. This option is available when the Image Conversion Format option is a value of JPEG, JPEG2000, or TIFF. The default is `Off`. The following values are valid.

- **Embed Profile:** Embed an ICC profile.
- **Off:** Do not embed an ICC profile.

**Grayscale Policy:**

The gray scale management to be applied to the output file and whether to embed an ICC profile. The default is `Off`. The following values are valid.

- **Embed Profile:** Embed an ICC profile.
- **Off:** Do not embed an ICC profile.

**Color Space:**

The color space management scheme to use. The default is `Determine Automatically`. The following values are valid.

- **RGB**: Use the Red, Green, and Blue color space typical for display on a screen.
- **CMYK**: Use the Cyan, Magenta, Yellow, and Key (Black) color space management typical for print.
- **Grayscale**: Use gray scale color space management.
- **Determine Automatically**: The color space to use is determined automatically.

**Resolution:**

The dots per inch (dpi) that specifies the resolution of the image. No default is provided. You can type any value, but the common values to use are 72, 96, 150, 300, 600, 1200, and 2400.

**Include Comments:**

Whether to preserve the appearance of comments in the resulting image file. The default is `True`, which means to preserve the appearance of the comments.

**Image Size Width:**

Sets the width of the image. No default value is provided. The default measurement unit is In. The valid values of measure that can be selected are in (inches), mm (millimeters), cm (centimeters), and in (points).

**Image Size Height:**

Sets the height of the image. No default value is provided. The default measurement unit is In. The valid values of measure that can be selected are in (inches), mm (millimeters), cm (centimeters), and in (points).

**Page Range:**

The page numbers and page ranges delimited by a comma. For example, to print page 22, pages 7 to 9, and page 13, type 2, 7-9, 13. No default is provided.

**Rows Per Strip:**

The number of rows of pixels per strip to use for encoding the TIFF image. This option is available when the Image Conversion Format option is a value of TIFF. A value greater than zero specifies the number of rows per strip. A value of `0` sets the rows per strip equal to the image length, resulting in a single strip. The default of `-1` sets the rows per strip equal to infinity, resulting in a single strip.

**Tile Size:**

The image tile size in pixels. This option is available when the Image Conversion Format option is a value JPEG2000. The default is `256`. Valid values are 128 - 2048 (pixels).

**Interlace:**

Specifies whether the image is interlaced. This option is available when the Image Conversion Format option is a value of PNG. The default is `None`. The following values are valid.

- **None:** Creates an image that displays in a web browser only after downloading is complete.
- **Adam7:**Creates an image that displays low-resolution versions in a browser while the full image file is downloading.

**PNG Filter:**

The filtering algorithm to use for the image. This option is available when the Image Conversion Format option is a value of PNG. The default is `Adaptive`. The following values are valid.

- **None:** Compresses the image without a filter. Recommended for indexed-color and bitmap-mode images.
- **Sub:** Optimizes the compression of images with even horizontal patterns or blends.
- **Up:** Optimizes the compression of images with even vertical patterns.

- **Average:** Optimizes the compression of low-level noise by averaging the color values of adjacent pixels.
- **Paeth:** Optimizes the compression of low-level noise by reassigning adjacent color values.
- **Adaptive:** Applies one of the filtering algorithms best suited for the image, such as Sub, Up, Average, or Paeth. Select Adaptive when it is uncertain which PNG filter to use.

**Monochrome Compression:**

The compression format to use. This option is available when the Image Conversion Format option is a value of TIFF. The default is `CCITT G4`. The following values are valid.

- **None:** Specifies not to use compression.
- **CCITT G3:** International Coordinating Committee for Telephony and Telegraphy Group 3 compression algorithm that compresses an image one row at a time.
- **CCITT G4:** International Coordinating Committee for Telephony and Telegraphy Group 4 compression algorithm that compresses an image one row at a time and, in general, produces the smallest file sizes.
- **LZW:** A table-based lookup compression algorithm.
- **ZIP:** A compression algorithm that is best suited for images with large areas of single colors or repeating patterns.

**Multipage TIFF:**

Sets whether the output TIFF should span multiple pages when converting a PDF file. This option is available when the Image Conversion Format option is TIFF. A value of `False` means that multiple image files are used for an individual page in the PDF file. The default is `True`, which means that a single TIFF image is returned for a single PDF file.

**Output properties**

Property for the output image file.

*Result*

The location to store the output image files. The data type is a `list` of `document` values.

For information about retrieving values from a list, see Accessingdata in data collections.

**Exceptions**

This operation can throw a ConvertPDFException exception.

## toPS2 operation

Converts a PDF document to a PostScript file.

For information about the General and Route Evaluation property groups, see Commonoperation properties.

### Input properties

Properties to specify the PDF document and PostScript level.

*PDF Document*

The `document` value representing the PDF document to be converted.

If you provide a literal value, clicking the ellipsis button displays the Select Asset dialog box. (See AboutSelect Asset.)

*Convert PostScript Options*

A `ToPSOptionsSpec`value that contains various preferences used to convert the PDF document to an image file. With the exception of the conversion format, which must be specified, default values for preferences are used if no value is specified.

If you provide a literal value, the following groups of options are available.

**General:**

The group of options to chose the resolution, page size, and printing range.

- **Resolution (DPI):** The resolution of the output in dots per inch (dpi). The default is `300`.
- **Page Range:** The page numbers and page ranges delimited by a comma. For example, to print page 22, pages 7 to 9, and page 13, type `2, 7-9, 13`. No default is provided.
- **Page Size:** The page size of the PostScript output. The default is `DetermineAutomatically`. The following are valid values:
  - **A2**, **A3**, **A4**, **A5**, **Envelop**, **Executive**, **Folio**, **Legal**, **Letter**, **Tabloid**
  - **DetermineAutomatically:**Specifies that the service determines the page size to use.
  - **Custom:** When selected, the height and width can be typed in the respective Height and Width boxes. The default measurement unit is Pt. The valid values of measure that can be selected are Pt (points), in. (inches), mm (millimeters), and cm (centimeters).

**Output:**

The group of options that specifies printing options.

- **Include Comments:** Sets whether to preserve the appearance of comments in the resulting PostScript file. A value of `True` means to preserve the appearance of comments in the resulting PostScript file. The default is `False`, which means not to preserve the appearance of comments in the resulting PostScript file.
- **Convert TrueType To Type 1**: Sets whether to convert TrueType fonts to Type 1 fonts in the resulting file. A value of `True` means to convert TrueType fonts to Type 1 fonts. The default is `False`, which means not to perform the conversion.
- **Font Inclusion:** The fonts that are to be included in the resulting PostScript file. The default is `Embedded Fonts`. The following are valid values.
  - **None:** The system fonts are used instead of the embedded fonts and referenced fonts.
  - **Embedded Fonts:** The fonts used are taken directly from the PDF file.

  - **Embedded Fonts and Referenced Fonts:** The fonts used are taken from the PDF file and the system in use.

- **Emit CIDFontType2:** Sets whether to preserve the hinting information in the original font in the resulting PostScript file. A value of `True` means to emit only CIDFontType2 fonts. The default is `False`, which means that the CIDFontType2 fonts are converted to CIDFontType0 fonts, resulting in compatibility with a wider range of printers.
- **Shrink to Fit:** Sets whether to shrink the contents of the input PDF file to fit the selected page size. A value of `True` means to shrink the contents of the input PDF file. The default is `False`,which means not to shrink the contents.
- **Expand to Fit:** Sets whether to expand the contents of the input PDF file to fit the selected page size. A value of `True` means to expand the contents of the PDF file to fit the selected page size. The default is `False`, which means not to expand the contents.
- **Rotate and Center:** Sets whether to rotate and center the contents of the input PDF file in the selected page. A value of `True` means to rotate and center the contents of the PDF file. The default is `False`, which means not to center or rotate the contents.
- **Reverse:** Sets whether to reverse the contents of the input PDF file in the page. A value of `True` means to reverse the contents of the input PDF File in the page. The default is `False`, which means not to reverse the contents.

**Marks and Bleeds:**

The group of options for the marks to add to the resulting PostScript file.

- **Trim Marks:** Sets whether to place a mark in each of the four corners to indicate the PDF trim box boundaries. A value of `True` means to include the trim marks. The default is `False`, which means not to include the trim marks.
- **Bleed Marks:** Sets whether to place a mark in each of the four corners to indicate the PDF bleed box boundaries. A value of `True` means to place a mark in each of the four corners to indicate the PDF bleed box boundaries. The default is `False`, which means not to place the bleed marks.
- **Registration Marks:** Sets whether to place a registration mark outside the crop area of the page. A value of `True` means to place a registration mark outside the crop area of the page. The default is `False`, which means not to place a registration mark.
- **Color Bars:** Sets whether to place a color bar at the top of the page, outside the crop area. A color bar shows a single box for each spot or process color in the document. A value of `True` specifies to place a color bar in the document. The default is `False`, which means not to place a color bar.
- **Page Information:** Sets whether to place page information outside the crop area of the page. A value of `True` specifies to place page information outside of the crop area of the page. The default is `False`, which means not place page information on the page.
- **Line Weight:** The weight of the lines used for the trim marks, bleed marks, and registration marks. The default is `0.25 pt`. The values that can be selected are `0.125 pt`, `0.25 pt`, and `0.50 pt`.
- **Style:** The style of the printer marks to create. The default is `Default`. The following values are valid.
  - **Default:** Adobe InDesign print marking conventions.
  - **InDesignJ1:** Adobe InDesign print marking conventions, first Japanese ID.

  - **InDesignJ2:**Adobe InDesign print marking conventions, second Japanese ID
  - **Illustrator:**Adobe InDesign print marking conventions.
  - **IllustratorJ:** Adobe InDesign print marking conventions, Japanese ID.
  - **QuarkXPress:** QuarkXpress print marking conventions.

**PostScript:**

The group of options to specify PostScript parameters.

- **PostScript:** The PostScript language compatibility. The default is `Language Level 2`. The following values are valid.
  - **Language Level 2:** The PostScript language that was released in 1992 supports color printing.
  - **Language Level 3:** The PostScript language that was released in 1997 supports more fonts, has better graphics handling, and includes several features to speed up PostScript printing.
- **Allow Binary Content:** Sets whether to create the PostScript file in binary format. The default is `False`, which means not to create it in binary format.
- **Use Maximum Available JPEG2000 Image Resolution:** Sets whether to use the highest resolution for printing JPEG 2000 images. The default is `True`, which means to use the highest resolution possible. If a value of `False` is chosen, the pixels per inch selected is the value configured in the Resolution option.
- **Emit PS Form Objects:** Sets whether to emit form objects for Form XObjects, which are used to represent complex objects that appear multiple times in the PDF file. A value of `True` means that the overall size of the print job may be reduced, but the printer memory used is increased. The default is `False`, which means not to emit the form objects.
- **Color:** Sets whether to use color and to produce one page of output per page. The default is `Composite`.
  - **Composite:**Produce a PostScript file that includes color.
  - **Composite Gray:**Produce a PostScript file that does not include color.

**Output properties**

*Output Postscript*

The location to store the converted PostScript file. The data type is `document`.

**Exceptions**

This operation can throw a ConvertPDFException exception.

## toSWF operation

Converts a PDF document to an SWF file. The resulting SWF files are returned as java.util.List of com.adobe.idp.Document.

For information about the General and Route Evaluation property groups, see Commonoperation properties.

### Input properties

Properties to specify the document to be converted and the conversion options.

*PDF Document*

The `document` value that represents the PDF document to be converted.

If you provide a literal value, clicking the ellipsis button displays the Select Asset dialog box. (See AboutSelect Asset.)

*Convert SWF Options*

A `ToSWFOptionsSpec` value that contains various preferences used to convert the PDF document to an SWF file. With the exception of the conversion format, which must be specified, default values for preferences are used if no value is specified.

The following preferences are available.

Height: Sets the height of the SWF file.

Width: Sets the width of the SWF file.

Version: Specifies the version with which the SWF file will be encoded.

Sample X: Sets the down-sampling factor of the image in x direction.

Sample Y: Sets the down-sampling factor of the image in y direction.

Process Signature: Specifies whether signed signature fields are processed during conversion.

**Page Range**: Specifies the pages to be converted.

### Output properties

Property for the output SWF file.

*Result*

The location to store the output SWF files. The data type is a `list` of `document` values.

For information about retrieving values from a list, see Accessingdata in data collections.

### Exceptions

This operation can throw ConvertPDFException or DSCException exception.

## Convert PDF exceptions

The Convert PDF service provides the following exceptions for throwing exception events.

### ConvertPDFException

Thrown if an error occurs during a Convert PDF operation.

**DSCException**

Thrown if an error occurs during a Convert PDF operation.

# 22.18. Decision Point

The Decision Point service provides the execute operation, which represents a point in the process where a decision is made that affects how the process progresses.

For information about using the Decision Point service, see Services Reference for AEM Forms

.

## execute

Use the execute operation when you need multiple routes to originate at an operation but no single step exists in the business process that requires the evaluation of multiple routes. The execute operation acts as a node in the process that serves as the origin of many routes, but has no executable function itself.

Several situations require the use of the execute operation:

- Several routes need to be evaluated to determine the first operation to execute in a process.

  *This situation occurs when a process is initiated when a person submits a form, and form data determines the first action to execute in the process map. For example, a customer can fill an invoice dispute form through your corporate web site. The dollar amount of the invoice determines whether the form is routed to a first-level manager for approval or to a credit representative for processing.*
- Several different routes in a process converge at a point where a set of rules are evaluated.

  *This situation can occur when the process loops to a step where a set of rules are reevaluated. For example, in a quality assurance process, an issue may have to go through a retesting process until it is fixed and the process can proceed. This situation can also occur if several branches converge after running in parallel. For example, in a process for hiring new employees, when an applicant is hired, several subprocesses are initiated as part of the hiring process. When each of the subprocesses completes, multiple rules based on the data of each subprocess are evaluated to determine the next step.*

For information about the General and Route Evaluation property groups, see Commonoperation properties.

# 22.19. Default Render

The service for the Default Render process that forms workflow provides. The service performs the following tasks:

- Retrieves a PDF form from a specified URL (usually from the repository)
- Enables the form to be used offline with Workspace
- Merges binary data with the form

- Enables the form to be used in Workspace

This service is the default configured render service for Document Form and xfaForm variables. It is recommended that you do not modify the Default Render process. If you want to use a similar process, you should make a copy of the process and modify the copy.

**IMPORTANT:** This process may be updated during upgrades of AEM Forms and therefore, you may need to update your version of the service.

## invoke operation

Retrieves and merges form data with the PDF form retrieved from the repository. The PDF form is modified for online and offline use with Workspace.

For information about the General and Route Evaluation property groups, see Commonoperation properties.

### Input properties

*Assigned User's Id*

A `string` value that specifies the unique identifier for the user who is assigned the task.

*Input Form Data*

A `document` value that represents the form data to merge with the PDF form.

If you provide a literal value, clicking the ellipsis button opens the Select Asset dialog box. (See About-Select Asset.)

*FormUrl*

A `string` value that specifies the URL location of the PDF form in the repository or at a network location on a file system.

*Task Route List*

A `string` value that contains the names of routes, delimited by a semicolon (;), that can be selected when completing a task. For example, a list of task routes can appear as the following value:

```
[rname1];[rname2];[rname3]
```

where *[rname]* represents the name of each route as defined in the process map.

*Reply Server Email*

A `string` value that specifies the email address to respond to. This email address is configured on the AEM Forms Server. (See forms workflow administration help

.)

*Submit Target Url*

A `string` value that specifies the URL of the web application or servlet that the submitted form data is sent to. For example, to access Workspace, type `http://`*[server name]*`:`*[port]*`/workspace-server/submit` where *[server name]* specifies the name of the AEM Forms Server and *[port]* specifies the port number. The following default port numbers are used by AEM forms for each application server.

- **JBoss:**`8080`
- **WebLogic:**`7001`
- **WebSphere:**`9080`

  **NOTE:** You can also use HTTP over Secure Socket Layer (SSL) for the Submit Target Url property.

*Task Id*

A `string` value that specifies the unique identifier for the task.

*Task Status*

A `string` value that specifies the run-time status of the task.

**Output properties**

*Output PDF Form*

The location to store the document that this operation creates. The data value is `document`.

**Invocation Policy properties**

Specifies whether to wait for a response after invoking the operation by selecting either Do Not Wait For Response or Wait For Response. The default is `Wait For Response`.

*Do Not Wait For Response*

Specifies that no response is required from the server.

*Wait For Response*

Specifies that a response is required from the server before further processing.

# 22.20. Distiller

The Distiller service creates PDF documents from PostScript, Encapsulated PostScript (EPS), and PRN files.

For information about using the Distiller service, see Services Reference for AEM Forms

.

## CreatePDF operation

Converts PostScript, Encapsulated PostScript (EPS), and printer text files (PRN) to PDF documents.

For information about the General and Route Evaluation property groups, see Commonoperation properties.

### Input properties

Properties to specify the input document to be converted and the conversion settings to use.

*Input Document*

The `document` value that represents the content to convert to a PDF document. It must be a valid PostScript or EPS document. No default is provided.

If you provide a literal value, clicking the ellipsis button displays the Select Asset dialog box. (See AboutSelect Asset.)

### Conversion options properties

*Adobe PDF settings*

(Optional) A `string` value that represents the PDF output settings to be applied. No default is provided, and the following values are valid:

- High Quality Print
- Oversized Pages
- PDFA1b 2005 CMYK
- PDFA1b 2005 RGB
- PDFA Draft
- PDFX1a 2001
- PDFX1a 2003
- PDFX3 2002
- PDFX3 2003
- Press Quality
- Smallest File Size
- Standard

  *When no value is specified, the service uses the default PDF settings that are configured in administration console.*

### Advanced options properties

*Filename with Extension*

(Optional) A `string` value that represents the file name, with the appropriate extension, of the PostScript or EPS document to be converted. No default is provided.

*Security Settings*

(Optional) A `string` value that specifies security settings. The single available option is No Security. When no value is specified, the service uses the default PDF settings that are configured in administration console.

*Settings Document*

(Optional) A `document` value that represents the file that contains settings to be applied while generating the PDF document, such as optimization for web view, as well as the settings that are applied after the PDF document is created, such as initial view and security.

If you provide a literal value, clicking the ellipsis button displays the Select Asset dialog box. (See AboutSelect Asset.)

*XMP Document*

(Optional) A `document` value that represents the file that contains metadata information to be applied to the generated PDF document. Only UTF-8 encoded XMP metadata is supported. The metadata should conform to the Adobe Extensible Metadata Platform (XMP) format. For details about the format and for the specifications, see XMP Utilities.

If you provide a literal value, clicking the ellipsis button displays the Select Asset dialog box. (See About-Select Asset.)

**Output properties**

Properties for specifying the output PDF document, operation results, and log files.

*Created Document*

The location to store the output PDF document. The data type is `document`.

**Optional Output properties**

*Log Document*

The location to store an optional log file. The log file is a plain text file that contains the output from the Distiller service. The log file is not generated under all circumstances. The presence of a log file is not an indication of an error in the conversion because a log file may contain only diagnostic messages. The data type is `document`.

**Exceptions**

This operation can throw ConversionException, FileFormatNotSupportedException, and InvalidParameterException exceptions.

## Distiller exceptions

The Distiller service provides the following exceptions for throwing exception events.

**ConversionException**

Thrown if a conversion fails because of an underlying exception.

**FileFormatNotSupportedException**

Thrown when a given file format is not supported.

**InvalidParameterException**

Thrown when a given parameter is not valid.

# 22.21. DocConverter

Validates and transforms a PDF to the PDF/A format.

For information about using the DocConverter service, see Services Reference for AEM Forms

.

## Validate PDF/A operation

Validates whether a PDF is compliant with the PDF/A format.

For information about the General and Route Evaluation property groups, see Commonoperation properties. In addition, this operation provides the following property groups and exceptions:

Input

Output

Exceptions

**Input properties**

Properties for specifying the PDF document and the options to use for PDF/A validation.

*Input PDF Document*

A `document` value that represents the PDF document to be validated as PDF/A.

If you provide a literal value, clicking the ellipsis button displays the Select Asset dialog box. (See AboutSelect Asset.)

*PDF/A Options*

(Optional) A `PDFAValidationOptionSpec` value that represents the options to be used for validation.

If you provide a literal value, specify the following values.

**Compliance:**

**Result level:**

The default is `PASS_FAIL`. The following values are valid.

- **PASS_FAIL:**A report contains the following information about the operation result:
  - Whether the operation was successful
  - Whether the PDF document is PDF/A compliant
  - Whether certified digital signatures are permitted
- **SUMMARY:**A report contains the following information about the operation result:
  - Whether the operation was successful
  - Whether the PDF document is PDF/A compliant
  - Whether certified digital signatures are permitted
  - A summarized count of any validation errors that occurred
- **DETAILED:** A report contains the following information about the operation result:
  - Whether the operation was successful
  - Whether the PDF document is PDF/A compliant
  - Whether certified digital signatures are permitted
  - A list of each error that occurred with specific information about each error

**Allow Certification Signatures:**

The default is `True`, which means that certification signatures are permitted.

**Ignore Unused Resources:**

Sets whether to perform validation on resources that are not used in the PDF document. The default is `True`, which means not to perform validation on unused resources.

**Job Log Level:**

Sets the log level for this operation. The default is `INFO`. The following values can be used.

**OFF:**

Turn off logging of messages on the server.

**SEVERE:**

Log messages only when they indicate a serious failure.

**WARNING:**

Log messages that indicate potential problems and serious failures.

**INFO:**

Log messages that are for informational purposes, indicate potential problems, and indicate serious failures.

**CONFIG:**

Log only static configuration messages.

**FINE:**

Log tracing information.

**FINER:**

Log fairly detailed tracing information.

**FINEST:**

Log highly detailed tracing information.

**NOTE:** The use of FINE, FINER, and FINEST will negatively impact performance on the server. Use them for debugging purposes only.

### Output properties

Property for specifying the result of the operation.

*PDF /Validation Result*

The location to store the results of the operation, which includes the Job log file, the converted PDF document, and the validation report. The data type is `com.adobe.livecycle.docconverter.client.PDFAValidationResult`.

### Exceptions

This operation can throw a ValidationExceptionexception.

## Convert to PDF/A operation

Converts a PDF document to PDF/A format using the options provided.

For information about the General and Route Evaluation property groups, see Commonoperation properties.

### Input properties

Properties for specifying the input DDX file and optional documents and environmental values.

*Input PDF Document*

A `document` value that represents the PDF document to be validated as PDF/A.

If you provide a literal value, clicking the ellipsis button displays the Select Asset dialog box. (See AboutSelect Asset.)

*PDF/A Conversion Options*

A `PDFAConversionOptionSpec` value that represents the options to be used for validation.

If you provide a literal value, specify the following values.

**Compliance:**

The compliance level for converting to PDF/A. The default is `PDFA_1B`, which is the minimal level for documents scanned from paper or microfiche sources.

**Result level:**

The level of reporting when the operation runs. The default is `PASS_FAIL`. The following values are valid.

**PASS_FAIL: A report contains the following information about the operation result:**

Whether the operation was successful

- Whether the PDF document is PDF/A compliant
- Whether certified digital signatures are permitted

**SUMMARY: A report contains the following information about the operation result:**

Whether the operation was successful

- Whether the PDF document is PDF/A compliant
- Whether certified digital signatures are permitted
- A summarized count of any validation errors that occurred

**DETAILED: A report contains the following information about the operation result:**

Whether the operation was successful

- Whether the PDF document is PDF/A compliant
- Whether certified digital signatures are permitted
- A list of each error that occurred with specific information about each error

**Signatures:**

Sets whether to archive a digital signature. The default is `ARCHIVE_AS_NEEDED`. The following values are valid.

**ARCHIVE_AS_NEEDED:**

Signatures are archived only when the signature can no longer be maintained in the PDF document.

**ARCHIVE_ALWAYS:**

Signatures are archived regardless of their validity in the resulting PDF/A.

**Color Space:**

The color space management scheme to use. The default is `S_RGB`. The following values are valid.

**S_RGB:**

Standard RGB IEC61966-2.1

**COATED_FOGRA27:**

Europe ISO Coated FOGRA27

**JAPAN_COLOR_COATED:**

Japan Color 2001 Coated

**SWOP:**

U.S. Web Coated (SWOP) v2

**Job Log Level:**

Sets the log level for this operation. The default is`INFO`. The following values can be used.

**OFF:**

Turn off logging of messages on the server.

**SEVERE:**

Log messages only when they indicate a serious failure.

**WARNING:**

Log messages that indicate potential problems and serious failures.

**INFO:**

Log messages that are for informational purposes, indicate potential problems, and indicate serious failures.

**CONFIG:**

Log only static configuration messages.

**FINE:**

Log tracing information.

**FINER:**

Log fairly detailed tracing information.

**FINEST:**

Log highly detailed tracing information.

**NOTE:** The use of FINE, FINER, and FINEST will negatively impact performance on the server. Use them for debugging purposes only.

#### Output properties

Property for specifying the result of the operation.

*PDF/A Conversion Result*

The location to store the results of the operation, which include the Job log file, the converted PDF document, and the conversion report. The data type is `PDFAConversionResult`.

#### Exceptions

This operation can throw a ConversionExceptionexception.

### DocConverter exceptions

The DocConverter service provides the following exceptions for throwing exception events.

#### ConversionException

Thrown if an error occurs during conversion of a PDF document to PDF/A format.

#### ValidationException

Thrown if an error occurs during validation of a PDF document.

**NOTE:** To receive an OperationException exception, ensure that the variable defined to contain the exception data is at least 150 characters long.

## 22.22. Email

Enables processes to receive and send email messages.

You can modify the default settings for the Email service by using one of the following methods:

- In Workspace, modify settings for the Email service in the Component view. (See Editing service configurations.)
- In the administration console, select Services > Applications and Services > Service Management and click the Email service. (See Applications and Services AdministrationHelp
- .)

For information about using the Email service, see Services Reference for AEM Forms

.

## Email service configuration

The Email service can be configured with default properties for connecting to an SMTP server for sending email messages, and to either a POP3 or IMAP server for receiving messages. (See Editing service configurations.)

When you configure the connection properties, the Email service operations inherit the property values. However, the default connection properties can be overridden using the properties of each operation.

The following properties can be configured for the Email service.

**SMTP Host:**

The IP address or URL of the SMTP server to use for sending email.

**SMTP Port Number:**

The port used to connect to the SMTP server.

**SMTP Authenticate:**

Select if user authentication is required to connect to the SMTP server.

**SMTP User:**

The user name of the user account to use to log in to the SMTP server.

**SMTP Password:**

The password that is associated with the SMTP user account.

**SMTP Transport Security:**

The security protocol to use for connecting to the SMTP server:

- Select None if no protocol is used (data is sent in clear text)
- Select SSL if Secure Sockets Layer protocol is used.
- Select TLS if Transport Layer Security is used.

**POP3/IMAP Host:**

The IP address or URL of the POP3 or IMAP server to use for sending email.

**POP3/IMAP Username:**

The user name of the user account to use to log into the POP3 or IMAP server.

**POP3/IMAP Password:**

The password that is associated with the POP3 or IMAP user account.

**POP3/IMAP Port Number:**

The port used to connect to the POP3 or IMAP server.

**POP3/IMAP:**

The protocol to use for sending and receiving email.

**Receive Transport Security:**

The security protocol to use for connecting to the SMTP server:

- Select None if no protocol is used (data is sent in clear text).
- Select SSL if Secure Sockets Layer protocol is used.
- Select TLS if Transport Layer Security is used.

## Receive operation

Receives email messages and attachments from either a POP3 or IMAP email server. You can save metadata about the email, as well as the message content. You can also set filters for email messages, and set properties about the email server and user account to use.

Email metadata and content can be saved in any location in the process data model, such as a variable or form field, that supports the type of data.

For information about the General and Route Evaluation property groups, see Commonoperation properties.

### Filter properties

Use Filter properties to selectively retrieve email messages.

*Filter Subject*

A `string` value that represents the filter for the email subject. Only email messages that contain the specified text in their subject line are retrieved. Specify no value to receive emails regardless of the email subject.

When entering this value as an XPath expression, surround the string with single quotation marks. For the literal value and template, the quotation marks are not required.

For example, to receive emails that contain “the quick brown fox” in the subject line, specify the following string values:

- XPath expression - ‘brown’
- literal value - brown
- template - brown

*Filter From*

A `string` value that represents the filter for the email sender’s email address. Only email messages that are sent from the specified email address are retrieved. Specify no value to receive emails from any email address.

### Folder Options

Use Folder Options properties to manage email storage on the email server.

*Folder*

A `string` value that represents the name of the folder from which to retrieve the email. Specify no value to retrieve email from the default folder. This property has no effect if you use a POP3 email server.

*Delete After Processing*

A `boolean` value that, if `True`, indicates that email is deleted from the server after it is retrieved. A value of `False` indicates that the email remains on the server.

### Retrieved Email Contents properties

Use Retrieved Email Contents properties to specify where to save the content of email messages.

*Subject*

The location to save the email subject. The data type is `string`.

*Body (Text)*

The location to save the email body as plain text. The data type is `string`.

*Body (HTML)*

The location to save the email body as HTML code. The Body data is `null` if the email is delivered in plain text format. The data type is `string`.

### From properties

Use From properties to save the sender's email address.

*From*

The location to save the sender's email address. The data type is `string`.

*Reply To*

The location to save the email address to use for replying to the email message. The sender can designate an email address to use for replying to the message that is different than the address used to send the message. The data type is `string`.

### Retrieved Email Attributes properties

Use Retrieved Email Attributes properties to save metadata about the email message.

*Size*

The location to save the size of the email, in bytes. The data type is `int`.

*Received Date*

The location to save the date that the email was received on the email server. The data type is `date`.

*Send Date*

The location to save the date that the email was sent. The data type is `date`.

*Importance*

The location to save the level of importance of the email. The data type is `string`.

*X-Priority*

The location to save the XPriority of the email. The data type is `string`.

*Message ID*

The location to save the unique identification of the email message. The data type is `string`.

**Attachments properties**

Use Attachments properties to specify how to store email attachments.

*Attachment Mime Filter*

A `list` value that contains `string` values which indicate the mime type of the attachments to retrieve. If no value is provided, all attachments are retrieved.

If you provide a literal value, clicking the ellipsis button opens the Select MIME Type For Filter dialog box. In the dialog box, select the MIME types of the attachments to retrieve and then click OK.

For information about retrieving values from a list, see Accessingdata in data collections.

*Attachment Count*

The location to save the number of files that are attached to the email message. The data type is `int`.

*Attachments Retrieved*

The location to save the number of attachments that were retrieved from the email message. This number can be different than the number of attachments that were sent in the email message if you have specified a MIME-type filter for the Attachment MIME Filter property. The data type is `int`.

*Map of Attachments*

The location to save the retrieved attachments. You need to store the data in a `map` variable that holds `document` values. The key for each document in the map is the corresponding attachment name.

You can also save attachments as `list` data if that data type is better suited to how you want to process the attachments later in the process. (See Listof Attachments.)

*List of Attachments*

The location to save the retrieved attachments. You need to store the data in a `list` variable that holds `document` values.

You can also save attachments as `map` data if that data type is better suited to how you want process the attachments later in the process. (See Mapof Attachments.)

**Connection Settings properties**

Use the Connection Settings properties to specify information required for connecting to the email server.

*Use Global Settings*

A `boolean` value that specifies whether to use the connection settings in the Email service configuration or the properties of the operation. A value of `True` causes the service configuration to be used. A value of `False` causes the properties of the operation to be used. The default value is `True`.

If you use a literal value to configure this property, select Use Global Settings to use the service configuration. To use the operation properties, clear Use Global Settings and provide values for the other Connection Settings properties.

*POP3/IMAP Host*

A `string` value that represents the IP address or URI of the POP3 or IMAP server. For example, you can use the name of the computer that is resolved by Domain Naming Scheme (DNS).

*POP3/IMAP Port Number*

An `int` value that represents the port number that the POP3 or IMAP server uses. If no value is specified, 110 is used for POP3 servers, and 143 is used for IMAP servers.

*POP3/IMAP Username*

A `string` value that represents the email address that is associated with the user account on the POP3 or IMAP server that is used to receive the email message.

*POP3/IMAP Password*

A `string` value that represents the password to use for authenticating with the POP3 or IMAP server. The password is associated by the user account that is associated with the value of the POP3/IMAP Username property.

If no value is provided, no authentication is attempted.

*POP3/IMAP*

A `string` value that represents the protocol to use for exchanging information with the POP3 or IMAP server. Valid values are `pop3` and `imap`. The value you use depends on the type of email server you are using.

*Receive Transport Security*

A `string` value that represents the transport security used when communicating with the POP3 or IMAP server. The following values are valid.

**None:**

Data is sent in clear text

**SSL:**

Secure Sockets Layer

**TLS:**

Transport Layer Security

The value you specify depends on how the email server is configured.

**Test properties**

Tools for testing the configuration of the Receive operation.

*Test Dialog*

Provides the Receive Tester dialog box that you can use to verify that the Receive operation is configured correctly. The test retrieves email messages from the email server, but does not cause the messages to be deleted from the server.

**NOTE:** The test cannot be performed if any property values are expressed using XPath expressions or process variables. At design-time, locations in the process data model are not yet associated with data.

To perform the test, click the ellipsis button which opens the dialog box, and then click Test. The results of the test appear in the dialog box.

**Exceptions**

The exception event attached to this operation can catch the MessageNotFoundException, ConnectionFailedException, and ReceiveMailFailedExceptionexceptions.

## Send With Document operation

Sends an email message that has a single attachment. To send an email with multiple attachments, you need to use the Send With Map Of Attachments operation. (See SendWith Map of Attachments.) Send With Document is easier to configure than Send With Map Of Attachments, which makes it useful if you need to send only one document.

For information about the General and Route Evaluation property groups, see Commonoperation properties.

### To Addresses properties

Use the To Addresses properties to specify email recipients. You can specify email addresses directly or import email addresses from a defined user list. To import addresses from a defined user list, click Import From User List, and then select a user list.

**NOTE:** If the selected user list contains any group, then the email addresses of the users and sub-groups inside the group are not imported. However, the email address of the group is imported.

*To*

A `string` value that represents one or more email addresses to send the email to. Separate email addresses with a comma.

*CC*

A `string` value that represents one or more email addresses to send a copy of the email to. Separate email addresses with a comma.

*BCC*

A `string` value that represents one or more email addresses to send a copy of the email to. The addresses are hidden from other recipients of the email. Separate email addresses with a comma.

### From Addresses properties

Use the From Addresses properties to specify the email address used to send the email message.

*From*

A `string` value that represents the email address that is associated with the email user account that is sending the email message.

*Reply To*

(Optional) A `string` value that represents the email address to which responses to the email are sent. If no value is specified, the email address specified for the From property is used.

*Return To*

(Optional) A `string` value that represents the email address to use as return address for the email envelope. If no value is specified, the email address specified for the From property is used.

### Contents properties

Use the Contents properties to specify the content of the email message.

*Subject*

A `string` value that represents the text to use for the email message subject.

*Message Format*

A `string` value that represents the format to use to format the message. Valid values are `TEXT` (plain text) and `HTML`. The default value is `TEXT`.

*Message*

A `string` value that represents the message body of the email message. The format of the text you provide should correspond with the value specified for the Message Format property. If the value of Message Format is HTML, the text you provide should include HTML markup.

To provide a literal value, you need to use the Message Body Editor dialog box to specify the message. (See AboutMessage Body Editor.) Click the ellipsis button to display the dialog box.

*Character Set Encoding*

A `string` value that represents the character set encoding to use on the subject and body of the email. No default is provided. When no value is provided, the default character set configured on the AEM Forms Server is used.

Provide a character set encoding value when your Workspace users need character set encoding that is different than the one used by the server. For example, if your server uses UTF-8 and your users need to read Japanese characters in email messages, set the character encoding to ISO-2022JP.

If you use a literal value to configure this property, select an encoding format from the list:

**NOTE:** The list of values depends on the operating system you use.

- Big5, Big5-HKSCS
- EUC-JP, EUC-KR
- GB18030, GB2312, GBK
- IBM-Thai, IBM00858, IBM01140, IBM01141, IBM01142, IBM01143, IBM01144, IBM01145, IBM001146, IBM001147, IBM001148, IBM001148, IBM037, IBM1026, IBM1047, IBM273, IBM277, IBM278, IBM280, IBM284, IBM285, IBM297, IBM420, IBM437, IBM500, IBM775, IBM850, IBM852, IBM855, IBM857, IBM860, IBM861, IBM862, IBM863, IBM864, IBM865, IBM866, IBM868, IBM869, IBM870, IBM871, IBM918
- ISO-2022-CN, ISO-2022-JP, ISO-2022-KR, ISO-8859-1, ISO-8859-13, ISO-8859-15, ISO-8859-2, ISO-8859-3, ISO-8859-4, ISO-8859-5, ISO-8859-6, ISO-8859-7, ISO-8859-8, ISO-8859-9
- JIS_X0201, JIS_X0201-1990
- KO18-R
- Shift-JIS
- TIS-620
- US-ASCII
- UTF16, UTF-!6BE, UTF16LE
- (Default) UTF-8
- windows-1250, windows-1251, windows-1252, windows-1253, windows-1254, windows-1255, windows-1256, windows-1257, windows-1258, 31j
- X-Big5-Solaris, Xeuc-jp-linux, x-EUC-TW

- x-eucJP-Open
- x-IBM1006, x-IBM1046, x-IBM1097, x-IBM1098, x-IBM1112, x-IBM1122, x-IBM1123, x-IBM1124, x-IBM1381, x-IBM1383, x-IBM33722, x-IBM737, x-IBM834, x-IBM856, x-IBM874, x-IBM875, x-IBM921, x-IBM922, x-IBM930, x-IBM933, x-IBM935, x-IBM937, x-IBM942, x-IBM942C, x-IBM943, x-IBM943C, x-IBM948, x-IBM949, x-IBM949C, x-IBM950, x-IBM964, x-IBM970
- x-ISCII91
- x-ISO-2022-CN-CNS, x-ISO-2022-CN-GB, x-is-8859-11
- x-JIS0208, x-JISAutoDetect
- x-Johab
- x-MacArabic, x-MacCentralEurope, x-MacCroatian, x-MacCyrillic, x-MacDingbat, x-MacGreek, x-MacHebrew, x-MacIceland, x-MacRoman, x-MacRomania, x-MacSymbol, x-MacThai, x-MacTurkish, x-MacUkraine
- x-MS950-HKSCS
- x-mswin-936
- x-PCK
- x-windows-50220, x-windows-50221, x-windows-874, x-windows-949, x-windows-950, xwindows-iso2022jp

### Attachments properties

Use the Attachments properties to specify a file to attach to the email message.

*Attachment Name*

A `string` value that represents the file name to use for the attachment. You should include the file name extension if the email receiver needs to associate the file with software to open it.

*Attachment*

An `object` value that represents the file to attach to the email message.

**IMPORTANT:** If you specify a `string`*value, such as a URL for the file or a file name, a document that includes the string is attached, and not the document itself.*

### Connection Settings properties

Use the Connection Settings properties to specify information required for connecting to the SMTP email server.

*Use Global Settings*

A `boolean` value that specifies whether to use the connection settings in the Email service configuration or the properties of the operation. A value of `True` causes the service configuration to be used. A value of `False` causes the properties of the operation to be used. The default value is `True`.

If you use a literal value to configure this property, select Use Global Settings to use the service configuration. To use the operation properties, clear Use Global Settings and provide values for the other Connection Settings properties.

*SMTP Host*

A `string` value that represents the IP address or URI of the email server. For example, you can use the name of the computer that is resolved by Domain Naming Scheme (DNS).

*SMTP Port Number*

An `int` value that represents the port number that the email server uses. If no value is specified, the value of `25` is used.

*SMTP Authenticate*

A `boolean` value that, if `True`, indicates that authentication with the email server is required for sending email. A value of `False` indicates that no authentication is required.

*SMTP User*

A `string` value that represents the email address that is associated with the user account on the email server that is used to send the email message.

*SMTP Password*

A `string` value that represents the password to use for authenticating with the email server. The password is associated by the user account that is associated with the value of the SMTP User property.

*SMTP Transport Security*

A `string` value that represents the transport security used when communicating with the email server. The following values are valid.

**None:**

Data is sent in clear text

**SSL:**

Secure Sockets Layer

**TLS:**

Transport Layer Security

The value you specify depends on how the email server is configured.

**Exceptions**

The exception event attached to this operation can receive the ConnectionFailedException, and SendMailFailedException exceptions.

## Send With Map of Attachments operation

Sends an email message that has multiple attachments. The files to attach are saved in a `map` variable.

For information about the General and Route Evaluation property groups, see Commonoperation properties.

### To Addresses properties

Use the To Addresses properties to specify who to send the email to. You can specify email addresses directly or import email addresses from a defined user list.

**NOTE:** If the selected user list contains any group, then the email addresses of the users and sub-groups inside the group are not imported. However, the email address of the group is imported.

*To*

A `string` value that represents one or more email addresses to send the email to. Separate email addresses with a comma.

*CC*

A `string` value that represents one or more email addresses to send a copy of the email to. Separate email addresses with a comma.

*BCC*

A `string` value represents one or more email addresses to send a copy of the email to, but the addresses are hidden from other recipients of the email. Separate email addresses with a comma.

### From Addresses properties

Use the From Addresses properties to specify the email address used to send the email message.

*From*

A `string` value that represents the email address that is associated with the email user account that is sending the email message.

*Reply To*

A `string` value that represents the email address to which responses to the email are sent. If no value is specified, the email address specified for the From property is used.

*Return To*

(Optional) A `string` value that represents the email address to use as return address for the email envelope. If no value is specified, the email address specified for the From property is used.

### Contents properties

Use the Contents properties to specify the content of the email message.

*Subject*

A `string` value that represents the text to use for the email message subject.

*Message Format*

A `string` value that represents the format to use to format the message. Valid values are `TEXT` (plain text) and `HTML`. The default value is `TEXT`.

*Message*

A `string` value that represents the message body of the email message. The format of the text you provide should correspond with the value specified for the Message Format property. If the value of Message Format is `HTML`, the text you provide should include HTML markup.

To provide a literal value, you need to use the Message Body Editor dialog box to specify the message. (See AboutMessage Body Editor.) Click the ellipsis button to display the dialog box.

*Character Set Encoding*

A `string` value that sets the character set encoding to use on the subject and body of the email. No default is provided. When no value is provided, the default character set configured on the AEM Forms Server is used.

Provide a character set encoding value when your Workspace users need character set encoding that is different than the one used by the server. For example, if your server uses UTF-8 and your users need to read Japanese characters in email messages, set the character encoding to ISO-2022JP.

If you use a literal value to configure this property, select an encoding format from the list:

**NOTE:** The list of values depends on the operating system you use.

- Big5, Big5-HKSCS
- EUC-JP, EUC-KR
- GB18030, GB2312, GBK
- IBM-Thai, IBM00858, IBM01140, IBM01141, IBM01142, IBM01143, IBM01144, IBM01145, IBM001146, IBM001147, IBM001148, IBM001148, IBM037, IBM1026, IBM1047, IBM273, IBM277, IBM278, IBM280, IBM284, IBM285, IBM297, IBM420, IBM437, IBM500, IBM775, IBM850, IBM852, IBM855, IBM857, IBM860, IBM861, IBM862, IBM863, IBM864, IBM865, IBM866, IBM868, IBM869, IBM870, IBM871, IBM918
- ISO-2022-CN, ISO-2022-JP, ISO-2022-KR, ISO-8859-1, ISO-8859-13, ISO-8859-15, ISO-8859-2, ISO-8859-3, ISO-8859-4, ISO-8859-5, ISO-8859-6, ISO-8859-7, ISO-8859-8, ISO-8859-9
- JIS_X0201, JIS_X0201-1990
- KO18-R
- Shift-JIS
- TIS-620
- US-ASCII
- UTF16, UTF-!6BE, UTF16LE
- (Default) UTF-8
- windows-1250, windows-1251, windows-1252, windows-1253, windows-1254, windows-1255, windows-1256, windows-1257, windows-1258, 31j
- X-Big5-Solaris, Xeuc-jp-linux, x-EUC-TW

- x-eucJP-Open
- x-IBM1006, x-IBM1046, x-IBM1097, x-IBM1098, x-IBM1112, x-IBM1122, x-IBM1123, x-IBM1124, x-IBM1381, x-IBM1383, x-IBM33722, x-IBM737, x-IBM834, x-IBM856, x-IBM874, x-IBM875, x-IBM921, x-IBM922, x-IBM930, x-IBM933, x-IBM935, x-IBM937, x-IBM942, x-IBM942C, x-IBM943, x-IBM943C, x-IBM948, x-IBM949, x-IBM949C, x-IBM950, x-IBM964, x-IBM970
- x-ISCII91
- x-ISO-2022-CN-CNS, x-ISO-2022-CN-GB, x-is-8859-11
- x-JIS0208, x-JISAutoDetect
- x-Johab
- x-MacArabic, x-MacCentralEurope, x-MacCroatian, x-MacCyrillic, x-MacDingbat, x-MacGreek, x-MacHebrew, x-MacIceland, x-MacRoman, x-MacRomania, x-MacSymbol, x-MacThai, x-MacTurkish, x-MacUkraine
- x-MS950-HKSCS
- x-mswin-936
- x-PCK
- x-windows-50220, x-windows-50221, x-windows-874, x-windows-949, x-windows-950, xwindows-iso2022jp

**Attachments properties**

Use the Attachments properties to specify the files to attach to the email message.

*Attachments*

A `map` value that contains the attachments for the email message, represented as `document` values or `string` values. If the values are of type `string`, the value represents either a URL or a file name:

- For a URL, the target of the URL is the content of the attachment. The AEM Forms Server needs to be able to access the URL target.
- For a file name, the path is either on the AEM Forms Server or other location that the AEM Forms Server can access.

  *If the AEM Forms Server is running in a Windows environment, file names need to be in the following format:*

  `file:///[file name]`

  *For example,* `file:///c:/attachment.pdf`*.*

The keys for the map are the attachment file names. The file names should include the file name extension if the email receiver needs to associate the file with software to open it. For information about retrieving values from a map, see Accessingdata in data collections.

To express the literal value of the property, you need to use the Attachments dialog box. (See AboutAttachments.)

### Connection Settings properties

Use the Connection Settings properties to specify information required for connecting to the SMTP email server.

*Use Global Settings*

A `boolean` value that specifies whether to use the connection settings in the Email service configuration or the properties of the operation. A value of `True` causes the service configuration to be used. A value of `False` causes the properties of the operation to be used. The default value is `True`.

If you use a literal value to configure this property, select Use Global Settings to use the service configuration. To use the operation properties, clear Use Global Settings and provide values for the other Connection Settings properties.

*SMTP Host*

A `string` value represents the IP address or URI of the email server. For example, you can use the name of the computer that is resolved by Domain Naming Scheme (DNS).

*SMTP Port Number*

An `int` value that represents the port number that the email server uses. If no value is specified, 25 is used.

*SMTP Authenticate*

A `boolean` value that, if `True`, indicates that authentication with the email server is required for sending email. A value of`False` indicates that no authentication is required.

*SMTP User*

A `string` value represents the email address that is associated with the user account on the email server that is used to send the email message.

*SMTP Password*

A `string` value represents the password to use for authenticating with the email server. The password is associated by the user account that is associated with the value of the SMTP User property.

*SMTP Transport Security*

A `string` value represents the transport security used when communicating with the email server. The following values are valid.

**None:**

Data is sent in clear text

**SSL:**

Secure Sockets Layer

**TLS:**

Transport Layer Security

The value you specify depends on how the email server is configured.

### Exceptions

The exception event attached to this operation can receive the ConnectionFailedException, and SendMailFailedException exceptions.

## Email exceptions

The Email service provides the following exceptions for receiving exception events.

### ConnectionFailedException

Thrown when connection with the mail server cannot be established.

### MessageNotFoundException

Thrown when no email messages are found on the mail server.

### ReceiveMailFailedException

Thrown when messages cannot be retrieved from the mail server.

### SendMailFailedException

Thrown when messages cannot be sent to the mail server.

## About Message Body Editor

Use the Message Body Editor dialog box to create the body text for email messages. You can use the dialog box for creating body text in both plain text and HTML formats.

- The text that you provide can include XPath expressions if part of the text is saved in a data location at run time.
- If the email message is in HTML format, you need to include the HTML markup. The dialog box provides several buttons that insert opening and closing tags for several commonly-used HTML elements.
- The dialog box shows a preview of the message in HTML format.

### Design tab

Use the Design tab to enter the message body text. The Design tab includes the following features:

- An editing area where you type the message text.
- The Message Format box, located at the bottom of the tab, that indicates the message format that the operation is configured to use.

  - **Text:** Indicates the message format is plain text.
  - **HTML:** Indicates the message format is HTML.

- A column of buttons, located along the right-side of the tab, that insert XPath expressions or HTML elements into the editing area.
  - **XPath:** Opens XPath Builder that you can use for creating XPath expressions that are inserted into the editing area.
  - **Para:** Inserts opening and closing `p` elements for defining a paragraph.
  - **Break:** Inserts a `br` element for defining a line break.
  - **Bold:** Inserts opening and closing `b` elements for bolding characters.
  - **Italics:** Inserts opening and closing `i` elements for italicizing characters.
  - **Color:**Inserts the following opening and closing `span` elements for defining text color:

    ```
    <span style="color:Red;"></span>
    ```

  - **Font:** Inserts the following opening and closing `span` elements for defining the font family and font size:

    ```
    <span style="font-family: arial, sans-serif;
    font-size: xx-small;"></span>
    ```

  - **Link:** Inserts the following opening and closing `a` elements for defining anchors:

    ```
    <a target="_blank" href=""></a>
    ```

- **Table:**Inserts the following opening and closing `table` elements, than defines a table with two rows and two columns.

  ```
  <table border="1" cellpadding="1" cellspacing="0" width="100%">
  <tr>
  <td align="left" valign="middle" width="50%"
  bgcolor="Silver"> </td>
  <td align="left" valign="middle" width="50%"
  bgcolor="Silver"> </td></tr>
  <tr>
  <td align="left" valign="middle" width="50%"> </td>
  <td align="left" valign="middle" width="50%"> </td></tr>
  </table>
  ```

### HTML Preview tab

Use the HTML Preview tab to see how the text in the Editing area of the Design tab is rendered as HTML. The preview is refreshed automatically when you click the HTML Preview tab.

## About Attachments

Use the Attachments dialog box to define file attachments to send with an email message using the Send With Map of Attachments operation. The dialog box includes a list that you populate with definitions of the file attachments.

To add a row to the list, click the + button. Click each cell in the row to specify values for each cell.

**Name:**

The file name. The file name should include the file name extension if the email receiver needs to associate the file with software to open it.

**Interpretation:**

Indicates how the operation should interpret the data that you specify as the file. Select one of the following values:

- XPath indicates that the file content is expressed as an XPath expression. The expression evaluates to the location in the process data model that stores the data to use as the file content.
- Literal indicates that the content is expressed as text. The text is the file content.
- Template indicates that the content is expressed as a template expression. The template is the file content. (See Specifyingtemplate expressions.)
- URL (XPath) indicates that the content is expressed as an XPath expression that evaluates to a data location that stores the URL of the file. The content is the target of the URL.
- URL (Literal) indicates that the content is expressed as text and is interpreted as the URL for the file. The content is the target of the URL.

**Value/XPath:**

The content of the file. The value that you specify must coincide with the value you specified for Interpretation. Click the ellipsis button to open XPath Builder, which you need to use to express the value.

# 22.23. Encryption

Interacts with encryption settings of PDF documents. You can encrypt a PDF document with a password to secure its contents or a certificate. You can encrypt the entire PDF document (includes its content, metadata, and attachments), everything other than its metadata, or only the attachments.

When you encrypt a PDF document with a password, you must specify two password values. The first password value, *open password*, enables the PDF document to be opened in Adobe Reader or Acrobat. The other password value, *permission password*, is used when removing password-based encryption from a PDF document.

When you encrypt a PDF document with a certificate, you must provide the location of the certificate.

Any combination of encrypting, certifying, and applying usage rights to the same document must occur in the following order. These services must be invoked within a short-lived process:

1) Encrypt (Encryption service or the Applypolicy (deprecated) operation of the Rights Management service)
2) CertifyPDF (Signature service)
3) ApplyUsage Rights (Acrobat Reader DC extensions service)

For information about using the Encryption service, see Services Reference for AEM Forms

.

## Certificate Encrypt PDF operation

Encrypts a PDF document by using a certificate.

For information about the General and Route Evaluation property groups, see Commonoperation properties.

### Input properties

Properties to specify a PDF document, certificate encryption identities, and the encryption parameters.

*Input PDF*

A `document` value that represents the PDF document to encrypt.

If you provide a literal value, clicking the ellipsis button opens the Select Asset dialog box. (See About-Select Asset).

*Certificate Encryption Identities*

A `list` of `Encryption Identity` values that represent the certificate encryption identities. A certificate identity can be any one of the following items:

- A certificate alias in trust store
- A certificate file on the file system
- A certificate file located at an LDAP URI. The LDAP URI must be an exact match on the LDAP server for proper retrieval of the certificate, which is used to encrypt the PDF document.

If you provide a literal value, certificates can be added or removed. Clicking Add Identity adds a new certificate encryption identity as an Identity entry to the pane below the Add Identity button. Clicking Remove Identity within an Identity entry removes an existing certificate encryption identity.

**NOTE:** Each certificate is represented as Identity [certificate] where certificate represents the position of the certificate in the list.

For each Identity entry, modify the options from the following option groups.

**Recipient:**

The location of the certificate used to encrypt the PDF document. No default is provided. Select one of the following options.

- **Select Alias:** The alias to a certificate located in trust store. When selected, type the name of the certificate alias.
- **Select Certificate File:** Specifies to use a certificate that is located at a network location or local computer. When selected, click the ellipsis button to open the Open dialog box where you can select a certificate from a local computer or a network location.

- **Select LDAP URI:** Specifies the location of the certificate. When the certificate is selected, type the URI of the LDAP server.

**Permissions:**

Specifies permissions to add to the encrypted PDF.

- **Restrict Printer And Editing Of The Document And Its Security Settings:** Sets the permissions for changing and printing a PDF document. Select this option to override the default permissions used to encrypt a PDF document by using a certificate.
- **Changes Allowed:** Select the changes that are allowed on the encrypted PDF document. The default is `Any Except Extracting Pages`. Select one of the following valid values.
- **None:** No changes are permitted on the encrypted PDF file.
- **Inserting, Deleting, and Rotating Pages:** Only inserting text, deleting, and rotating the page layout is permitted.
- **Form Fill and Sign Signature Fields:** Only filling in fields and digitally signing the PDF file is permitted.
- **Comments, Form Fill and Sign Signature Fields:** Only adding comments, filling in the fields in the PDF file, and digitally signing the PDF file is permitted.
- **Any Except Extracting Pages:** All editing of a PDF file is permitted except for extracting PDF pages from the
- **Printing Allowed:** Select whether printing is allowed. The default is `High Resolution`. Select one of the following valid values.
- **None:** Specifies that printing of the PDF file is not allowed.
- **High Resolution:** Specifies that printing of the PDF is permitted in high resolution.
- **Low Resolution (150 dpi):** Specifies that printing of the PDF file is permitted in low resolution up to 150 dpi.

*Certificate Encryption Option Spec*

A `Certificate Encryption Option Spec` value that represents encryption preferences that are used when encrypting a PDF document with a certificate. You can encrypt the entire PDF file (includes the metadata and attachments), everything except for the PDF file's metadata, or only the attachments.

If you provide a literal value, specify the following options in the PDF Encryption Options.

**Compatibility:**

The compatibility with earlier versions of Acrobat and Adobe Reader. Select one of the following values.

- **Acrobat 6.0 And Later:** Compatible with Acrobat 6 and later versions that encrypt PDF documents by using a 128-bit RC4 algorithm and provides fine-grained control over the operations that are permitted.
- **Acrobat 7.0 And Later:** Compatible with Acrobat 7 and later versions that encrypt PDF documents by using Advanced Encryption Standard (AES).

- **Acrobat 9.0 And Later:** Compatible with Acrobat 9 and later versions that encrypt PDF documents by using FIPs compliant AES 256 algorithm.

  The default is `Acrobat7.0 And Later`.

**All Document Contents:**

Select to encrypt the entire PDF document, including its attachments and metadata.

**All Contents Except Metadata:**

Select to encrypt all PDF document resources except for its metadata. This option is available only when Acrobat 6.0 And Later, Acrobat 7.0 And Later, or Acrobat 9.0 And Later is selected for the Compatibility option.

**Attachments Only:**

Select to encrypt only PDF document attachments. This option is available only when Acrobat 7.0 And Later or Acrobat 9.0 And Later is selected for the Compatibility option.

### Output properties

Property to specify an output value that represents PDF document that was encrypted by using a certificate.

*Output PDF*

The location to store the PDF document that was encrypted by using a certificate. The data type is `document`.

### Exceptions

This operation can throw an EncryptionServiceException exception.

## Get PDF Encryption Type operation

Returns the type of security that is used to secure a PDF document. A PDF document can be secured by encrypting it using a password or certificate, Rights Management, or another encryption mechanism.

For information about the General and Route Evaluation property groups, see Commonoperation properties.

### Input properties

Property to specify a PDF document.

*Input PDF*

A `document` value that represents an encrypted document.)

If you provide a literal value, clicking the ellipsis button opens the Select Asset dialog box. (See About-Select Asset.)

### Output properties

Property to specify an output value representing the password encryption.

*Encryption Type*

The location to store the value that specifies the type of security that was used to secure a PDF document. The data type is `EncryptionTypeResult`.

### Exceptions

This operation can throw an EncryptionServiceException exception.

## Password Encrypt PDF operation

Encrypts a PDF document by using a password.

For information about the General and Route Evaluation property groups, see Commonoperation properties.

### Input properties

Properties to specify a PDF document and the encryption parameters.

*Input PDF*

A `document` value that represents the PDF document to encrypt.

If you provide a literal value, clicking the ellipsis button opens the Select Asset dialog box. (See About-Select Asset.)

*Encryption Options*

A `Password Encryption Option Spec` value that specifies encryption settings. If you provide a literal value, specify the following options.

**Compatibility:**

Specifies the compatibility with earlier versions of Acrobat and Adobe Reader. The following values are valid.

- **Acrobat 3.0 And Later:** Compatible with Acrobat 3 and later versions that provide 40-bit encryption.
- **Acrobat 5.0 And Later:** Compatible with Acrobat 5 and later versions that provide 128-bit encryption.
- **Acrobat 6.0 And Later:** Compatible with Acrobat 6 and later versions that provide 128-bit encryption and also a finer granularity of control over what operations are permitted.
- **Acrobat 7.0 And Later:** Compatible with Acrobat 7 and later versions that provide Advanced Encryption Standard (AES).

- **Acrobat 9.0 And Later:** Compatible with Acrobat 9 and later versions that encrypt PDF documents by using FIPs compliant AES 256 algorithm.

  The default is `Acrobat 5.0 And Later`.

**All Document Contents:**

Select to encrypt the entire PDF document, including its attachments and metadata.

**All Contents Except Metadata:**

Select to encrypt all PDF document resources except for its metadata. This option is available only when Acrobat 6.0 And Later, Acrobat 7.0 And Later, or Acrobat 9.0 And Later is selected for the Compatibility option.

**Attachments Only:**

Select to encrypt only PDF document attachments. This option is available only when Acrobat 7.0 And Later or Acrobat 9.0 And Later is selected for the Compatibility option.

**Document Open Password:**

The `string` value that represents the password required to open a password-encrypted PDF document.

If you do not specify a value for the open password, users are not prompted for the open password value when opening the PDF document in Adobe Reader or Acrobat. However, users are prompted for the permission password value when they attempt to change settings in Acrobat or when removing encryption from the PDF document.

The value of the Document Open Password property must be different than the value of the Permissions Password property. If these properties are the same, an exception is thrown when the operation executes.

**Restrict Edit Of Security Settings:**

Determines whether permissions password is used. When selected, the Permissions Password, Printing Allowed, and Changes Allowed options can be modified. Otherwise, no permissions password can be specified, and defaults for the Printing Allowed and Changes Allowed options are used.

- **Permissions Password:** The password value that is required to modify permissions on the document or remove encryption.

  *If you specify a null value for the permission password, users are prompted for the open password value when opening the PDF document in Adobe Reader or Acrobat. However, users are not prompted for the permission password value when they attempt to change settings in Acrobat or when removing encryption from the PDF document. The value of the Permissions Password property must be different than the value of the Document Open Password property. If these properties are the same, an exception is thrown when the operation executes.*
- **Printing Allowed:** The permission that specifies whether printing is allowed. The default is `High Resolution`. The following values are valid.
- **None:** Specifies that printing of the PDF file is not allowed.

- **High Resolution:** Specifies that printing of the PDF is permitted in high resolution.
- **Low Resolution (150 dpi):** Specifies that printing of the PDF file is permitted in low resolution up to 150 dpi.
- **Changes Allowed:** The changes that are allowed on the PDF document. The default is `Any Except Extracting Pages`. The following values are valid.
- **None:** No changes are permitted on the encrypted PDF file.
- **Inserting, Deleting, And Rotating Pages:** Only inserting text, deleting, and rotating the page layout is permitted.
- **Form Fill And Sign Signature Fields:** Only filling in fields and digitally signing the PDF file is permitted.
- **Comments, Form Fill And Sign Signature Fields:** Only adding comments, filling in the fields in the PDF file, and digitally signing the PDF file is permitted.
- **Any Except Extracting Pages:** All editing of a PDF file is permitted except for extracting PDF pages from the document.

### Output properties

Property to specify a PDF document that was encrypted using a password.

*Output PDF*

The location to store the PDF document that was encrypted using a password. The data type is `document`.

### Exceptions

This operation can throw an EncryptionServiceException exception.

## Remove PDF Certificate Encryption operation

Returns an unsecured PDF document when given a PDF document encrypted with a certificate. After certificate encryption security is removed from a PDF document, it is no longer secure.

For information about the General and Route Evaluation property groups, see Commonoperation properties.

### Input properties

Properties to specify a PDF document and certificate alias.

*Input PDF*

A `document` value that represents a PDF document that is encrypted with a password.

If you provide a literal value, clicking the ellipsis button opens the Select Asset dialog box. (See About-Select Asset).

*Credential Alias*

A `string` value that represents the name of the local credential alias used to remove certificate encryption security from the PDF document.

The name of the local credential alias must be present and is managed by using Trust Store Management. (See Trust Store Management Help

.)

**Output properties**

Property to specify an output value that represents the unsecured PDF document.

*Output PDF*

The location to store the unsecured PDF document. The data type is `document`.

**Exceptions**

This operation can throw an EncryptionServiceException exception.

## Remove PDF Password Encryption operation

Returns an unsecured PDF document when given a password-encrypted PDF document and the permission password that is required to remove encryption from the PDF document. After password-based encryption is removed from a PDF document, it is no longer secure. You cannot use the open password value to remove encryption from a PDF document.

For information about the General and Route Evaluation property groups, see Commonoperation properties.

**Input properties**

Properties that represent the encrypted PDF document the permission password.

*Input PDF*

A `document` value that represents a PDF document encrypted by using a password.

If you provide a literal value, clicking the ellipsis button opens the Select Asset dialog box. (See About-Select Asset).

*Password*

A `string` value that represents the permission password used to allow removal of encryption from a PDF document.

**Output properties**

Property to specify an output value that represents the unsecured PDF document.

*Output PDF*

The location to store the unsecured PDF document. The data type is `document`.

**Exceptions**

This operation can throw an EncryptionServiceException exception.

## Unlock Certificate Encrypted PDF operation

Unlocks a PDF document when given a PDF document encrypted with a certificate. A PDF document must be unlocked in order for service operations to be performed on it.

**NOTE:** Use this operation in short-lived processes to perform additional operations on the unlocked PDF document.

For information about the General and Route Evaluation property groups, see Commonoperation properties.

**Input properties**

Properties that represent the encrypted PDF document and certificate.

*Input PDF*

A `document` value that represents a PDF document encrypted with a certificate to unlock.

If you provide a literal value, clicking the ellipsis button opens the Select Asset dialog box. (See About-Select Asset.)

*Credential Alias*

A `string` value that represents the name of the local credential alias used to remove certificate encryption security from the PDF document.

The name of the local credential alias must be present and is managed by using Trust Store Management. (See Trust Store Management Help

.)

**Output properties**

Property to specify an output value that represents the unlocked PDF document.

*Output PDF*

The location to store the unlocked PDF document. The data type is `document`.

**Exceptions**

This operation can throw an EncryptionServiceException exception.

## Unlock Password Encrypted PDF operation

Returns an unlocked PDF document when given a password-encrypted PDF document as input. A PDF document that is encrypted by using a password must be unlocked for service operations to be performed on it.

**NOTE:** Use this operation in short-lived processes to perform additional operations on the unlocked PDF document.

For information about the General and Route Evaluation property groups, see Commonoperation properties.

### Input properties

Properties that represent the encrypted PDF document and password.

*Input PDF*

A `document` value that represents a PDF document encrypted with a password to unlock.

If you provide a literal value, clicking the ellipsis button opens the Select Asset dialog box. (See About-Select Asset.)

*Password*

A `string` value that represents the password used to open a PDF document that is encrypted with a password.

### Output properties

Property to specify an output value that represents the unlocked PDF document.

*Output PDF*

The location to store the unlocked PDF document. The data type is `document`.

### Exceptions

This operation can throw an EncryptionServiceExceptionexception.

## Encryption exceptions

The Encryption service provides the following exceptions for throwing exception events.

### EncryptionServiceException

Thrown if an error occurs during the operation.

# 22.24. Execute Script

The Execute Script service lets you execute scripts in processes.

For information about the methods that the `patExecContext` object provides, see patExecContext reference.

For information about using the Execute Script service, see Services Reference for AEM Forms

.

## executeScript

Executes a script. Java is the supported scripting language for which BeanShell 1.3.0 is used as the implementation. For information about writing Java code for BeanShell, go tohttp://www.beanshell.org

.

The `patExecContext` object is implicitly available to your script. This object provides access to the process data model and is useful for manipulating data, such as variable values. Although you can also use the Set Value service to manipulate data, the executeScript operation is more flexible. For example, you can use a `for` loop in a script to populate a list value with documents when the number of documents is unknown until run time.

The following example creates a map of URLs that are used to call a custom web application.

```
//import the classes that the script references
import java.util.List;
import java.lang.String;
import java.util.Map;
import java.util.HashMap;

//get a list of file names that are stored in the process variable named
files
List fileNames =
patExecContext.getProcessDataListValue("/process_data/files");

//create a Map object for storing server URLs
Map outbound = new HashMap();

//get the URL of the web server, stored in the process variable named
serverURL
String webServerRoot =
patExecContext.getProcessDataStringValue("/process_data/@serverURL");

//for each file name, append the name to the URL as a parameter
//store the result in outbound
for (int i=0;i<fileNames.size();i++){
String currentFileName=(String)fileNames.get(i);
outbound.put(currentFileName, webServerRoot+"?doc="+ currentFileName);
}
```

```
//save the outbound map in the process variable named serverCalls
patExecContext.setProcessDataMapValue("/process_data/serverCalls",outbou
nd);
```

For information about the General and Route Evaluation property groups, see Commonoperation properties.

**Input operation**

*Script*

The script to execute.

## patExecContext reference

This section describes the methods that the `patExecContext` object provides.

**getProcessId**

Gets the identifier of the process instance that is currently being executed.

*Syntax*

```
getProcessId()
```

*Returns*

A `long` value that contains the process identifier.

**getBranchId**

Gets the identifier of the branch instance that is currently being executed.

*Syntax*

```
getBranchId()
```

*Returns*

A `long` value that contains the branch identifier.

**getActionId**

Gets the identifier of the operation instance that is currently being executed.

*Syntax*

```
getActionId()
```

*Returns*

A `long` value that contains the operation instance identifier.

**getExecuteTemplate**

Gets the action to be executed and casts it to an ExecuteTemplate.

*Syntax*

```
public ExecuteTemplate getExecuteTemplate()
```

**getActionTemplate**

Gets the action to be executed.

*Syntax*

```
public ActionTemplate getActionTemplate()
```

See also

com.adobe.workflow.template.document.ActionTemplate[Should this be a link?]

**getProcessDataBooleanValue**

Gets the process data value from the specified path as a `boolean` value.

*Syntax*

```
getProcessDataBooleanValue(aPath)
```

*Parameters*

`aPath` is a `java.lang.String` value that contains the XPath expression that evaluates to the `boolean` value in the process data model.

*Returns*

A `boolean` value coerced from the specified expression that represents a process data path.

**getProcessDataShortValue**

Gets the process data value from the specified path as a `short` value.

*Syntax*

```
getProcessDataShortValue(aPath)
```

*Parameters*

`aPath` is a `java.lang.String` value that contains the XPath expression that evaluates to the `short` value in the process data model.

*Returns*

A `short` value coerced from the specified expression that represents a process data path.

**getProcessDataIntValue**

Gets the process data value from the specified path as an `int` value.

*Syntax*

```
getProcessDataIntValue(aPath)
```

*Parameters*

`aPath` is a `java.lang.String` value that contains the XPath expression that evaluates to the `int` value in the process data model.

*Returns*

An `int` value coerced from the specified expression that represents a process data path.

**getProcessDataLongValue**

Gets the process data value from the specified path as a `long` value.

*Syntax*

```
getProcessDataLongValue(aPath)
```

*Parameters*

`aPath` is a `java.lang.String` value that contains the XPath expression that evaluates to the `long` value in the process data model.

*Returns*

A `long` value coerced from the specified expression that represents a process data path.

**getProcessDataDoubleValue**

Gets the process data value from the specified path as a `double` value.

*Syntax*

```
getProcessDataDoubleValue(aPath)
```

*Parameters*

`aPath` is a `java.lang.String` value that contains the XPath expression that evaluates to the `double` value in the process data model.

*Returns*

A `double` value coerced from the specified expression that represents a process data path.

**getProcessDataFloatValue**

Gets the process data value from the specified path as a `float` value.

*Syntax*

```
getProcessDataFloatValue(aPath)
```

*Parameters*

`aPath` is a `java.lang.String` value that contains the XPath expression that evaluates to the `float` value in the process data model.

*Returns*

A `float` value coerced from the specified expression that represents a process data path.

**getProcessDataStringValue**

Gets the process data value from the specified path as a `String` value.

*Syntax*

```
getProcessDataStringValue(aPath)
```

*Parameters*

`aPath` is a `java.lang.String` value that contains the XPath expression that evaluates to the `string` value in the process data model.

*Returns*

A `string` value coerced from the specified expression that represents a process data path.

**getProcessDataDocumentValue**

Gets the process data value from the specified path as a `Document` value.

**TIP:** You can also use the `getProcessDataValue` method to return an object and then determine if the object is of type `com.adobe.idp.Document`.

*Syntax*

```
getProcessDataDocumentValue(aPath)
```

*Parameters*

`aPath` is a `java.lang.String` value that contains the XPath expression that evaluates to the `string` value in the process data model.

*Returns*

A `document` value coerced from the specified process data path expression. If no `Document` object is found, an empty `Document` object is returned.

**getProcessDataListValue**

Gets the process data `list` value from the specified path as a `java.util.List` value.

**TIP:** You can also use the `getProcessDataValue` method to return an object and then determine if the object is of type `java.util.List`.

*Syntax*

```
getProcessDataListValue(aPath)
```

*Parameters*

`aPath` is a `java.lang.String` value that contains the XPath expression that evaluates to the `list`value in the process data model.

*Returns*

A `java.util.List` value coerced from the specified process data path expression. If no `List` object is found, an empty `List` object is returned.

**getProcessDataMapValue**

Gets the process data `map` value from the specified path as a `java.util.Map` value.

**TIP:** You can also use the `getProcessDataValue`*method to return an object and then determine if the object is of type java.util.Map.*

*Syntax*

```
getProcessDataMapValue(aPath)
```

*Parameters*

`aPath` is a `java.lang.String` value that contains the XPath expression that evaluates to the `map`value in the process data model.

*Returns*

A `java.util.Map` value coerced from the specified process data path expression. If no `Map` object is found, an empty `Map` object is returned.

**getProcessDataValue**

Gets the process data value from the specified path as an `Object` value.

*Syntax*

```
getProcessDataValue(aPath)
```

*Parameters*

`aPath` is a `java.lang.String` value that contains the XPath expression that evaluates to a value in the process data model.

*Returns*

An `object` value coerced from the specified process data path expression.

**setProcessDataBooleanValue**

Sets a `boolean` value in the process data model.

*Syntax*

```
setProcessDataBooleanValue( aPath, aVal)
```

*Parameters*

`aPath` is a `string` value that contains the XPath expression that evaluates to a location in the process data model in which to store the value.

`aVal` is a `boolean` value of `True` or `False`.

**setProcessDataShortValue**

Sets a `short` value in the process data model.

*Syntax*

```
setProcessDataShortValue(aPath, aVal)
```

*Parameters*

`aPath` is a `java.lang.String` value that contains the XPath expression that evaluates to a location in the process data model in which to store the value.

`aVal` is a `short` value.

**setProcessDataIntValue**

Sets an `int` value in the process data model.

*Syntax*

```
setProcessDataIntValue(aPath, aVal)
```

*Parameters*

`aPath` is a `java.lang.String` value that contains the XPath expression that evaluates to a location in the process data model in which to store the value.

`aVal` is an `int` value.

**setProcessDataLongValue**

Sets a `long` value in the process data model.

*Syntax*

```
setProcessDataLongValue(aPath, aVal)
```

*Parameters*

`aPath` is a `java.lang.String` value that contains the XPath expression that evaluates to a location in the process data model in which to store the value.

`aVal` is a `long` value.

**setProcessDataDoubleValue**

Sets a `double` value in the process data model.

*Syntax*

```
setProcessDataDoubleValue(aPath, aVal)
```

*Parameters*

`aPath` is a `java.lang.String` value that contains the XPath expression that evaluates to a location in the process data model in which to store the value.

`aVal` is a `double` value.

**setProcessDataListValue**

Sets a `list` value in the process data model.

*Syntax*

```
setProcessDataListValue(aPath, aVal)
```

*Parameters*

`aPath` is a `java.lang.String` value that contains the XPath expression that evaluates to a location in the process data model in which to store the value.

`aVal` is a `list` value.

### appendToProcessDataListValue

Appends the items in a `List` value to a process variable of type `list`.

*Syntax*

```
appendToProcessDataListValue(variableName, aVal)
```

*Parameters*

`variableName` is a `java.lang.String` that contains the name of the `list` variable to append to.

`aVal` is a `List` value. The items in `aVal` are appended to the `list` that `variableName` represents.

### setProcessDataListValue

Set the value of an item at a given position in a `list` variable.

*Syntax*

```
setProcessDataListValue(variableName, position, value)
```

*Parameters*

`variableName` is a `java.lang.String` that contains the name of the `list` variable.

`position` is an `int` value that contains the position in the `list` variable that is being set.

`value` is a `java.lang.Object` value to set in the `list` variable.

### removeProcessDataListValue

Removes the object from a given position in a `list` variable.

*Syntax*

```
removeProcessDataListValue(variableName, position)
```

*Parameters*

`variableName` is a `java.lang.String` value that contains the name of the `list` variable.

`position` is an `int` value that contains the position in the `list` variable that is being removed.

### setProcessDataMapValue

Set a `map` value in the process data model.

*Syntax*

```
setProcessDataMapValue(aPath, aVal)
```

*Parameters*

`aPath` is a `java.lang.String` value that contains the XPath expression that evaluates to a location in the process data model in which to store the value.

`aVal` is a `map` value.

**appendProcessDataMapValue**

Appends the item in a Map object to a process variable of type map.

*Syntax*

```
appendProcessDataMapValue(variableName, Map aVal)
```

*Parameters*

`variableName` is a `java.lang.String` value that contains the name of the `map` variable.

`aVal` is a `Map` value to append.

**setProcessDataMapValue**

Sets the value in a process data variable of type `map` for the specified key.

*Syntax*

```
setProcessDataMapValue(variableName, key, value)
```

*Parameters*

`variableName` is a `java.lang.String` value that contains the name of the `map` variable.

`key` is a `java.lang.String` value that contains the `key` in the `map` that is being set.

`value` is a `java.lang.Object` value to set for the `key` in the `map`.

**removeProcessDataMapValue**

Removes the object for a given `key` value from the variable of type `map`.

*Syntax*

```
removeProcessDataMapValue(variableName, key)
```

*Parameters*

`variableName` is a `java.lang.String` value that contains the name of the `map` variable.

`key` is a `java.lang.String` value that contains the `key` in the map for the value that is being removed.

### setProcessDataFloatValue

Sets a `float` value in the process data model.

*Syntax*

```
setProcessDataFloatValue(aPath, aVal)
```

*Parameters*

`aPath` is a `java.lang.String` value that contains the XPath expression that evaluates to a location in the process data model in which to store the value.

`aVal` is a `float` value.

### setProcessDataStringValue

Sets a `String` value in the process data model.

*Syntax*

```
setProcessDataStringValue(aPath, aVal)
```

*Parameters*

`aPath` is a `java.lang.String` value that contains the XPath expression that evaluates to a location in the process data model in which to store the value.

`aVal` is a `String` value.

### setProcessDataDocumentValue

Sets a `com.adobe.idp.Document` object in the process data model.

*Syntax*

```
setProcessDataDocumentValue(aPath, aVal)
```

*Parameters*

`aPath` is a `java.lang.String` value that contains the XPath expression that evaluates to a location in the process data model in which to store the value.

`aVal` is a `com.adobe.idp.Document` object.

### setProcessDataValue

Sets an `Object` value in the process data model.

*Syntax*

```
setProcessDataValue(aPath, aVal)
```

*Parameters*

`aPath` is a `java.lang.String` value that contains the XPath expression that evaluates to a location in the process data model in which to store the value.

`aVal` is an `Object value`.

**setProcessDataWithExpression**

Sets the value of a location in the process data model using the value that an XPath expression evaluates to.

*Syntax*

```
setProcessDataWithExpression(aPath, aExpression)
```

*Parameters*

`aPath` is a `java.lang.String` value that contains the XPath expression that evaluates to a location in the process data model in which to store the value.

`aExpression` is a `java.lang.String` value that represents the XPath expression.

**getContainedDataTypeForCollection**

Retrieves the data type that a `list` or `map` value contains.

*Syntax*

```
getContainedDataTypeForCollection(variableName)
```

*Parameters*

`variableName` is a java.lang.String that represents the name of the list or map variable.

*Returns*

An `int` value that denotes the type of object:

**0:**

boolean

**1:**

short

**2:**

int

**3:**

long

**4:**

float

**5:**

double

**6:**

decimal

**7:**

string

**8:**

binary

**9:**

date

**10:**

date_time

**11:**

xml

**13:**

document

**getProcessDataValueByVariableName**

Returns the value for a process variable that can be referenced by name.

*Syntax*

```
getProcessDataValueByVariableName(aVariableName)
```

*Parameters*

`aVariableName` is a `java.lang.String` value that represents the variable name.

*Returns*

An `Object` value that represents the value that the variable stores.

*Throws*

`VariableNotFoundException` exception if the named variable cannot be found.

# 22.25. File Utilities

Enables processes to interact with the file system of the AEM Forms Server or other file system that the server can access. Processes can use this service to perform the following tasks:

- Save data as files on the file system
- Retrieve information from files and save it as process data
- Manipulate directories and files on the file system

If you specify local paths to files or directories for any operation properties, the paths are interpreted as being on the file system of the AEM Forms Server.

**NOTE:** The user account that is used to run the AEM forms Server must have the required permissions to interact with the files and file locations that the service's operations target.

Most file operations are system-dependent and invoke operating system libraries. The behavior of the service operations can depend on the operating system that the AEM Forms Server runs on. Also, all file and directory paths are system-dependent.

For information about using the File Utilities service, see http://www.adobe.com/go/learn_aemforms_-services_61

.

For some of the operations listed, you must specify file or folder paths as input or output parameters. Use absolute file or folder paths where required.

- Examples of absolute file paths are:
  - Windows: `C:\Output\Sample-xslt.html`
  - UNIX: `/usr/yourUserName/Output/Sample-xslt.html`
- Examples of absolute folder paths are:
  - Windows: `C:\Output\`
- UNIX: `/usr/yourUserName/Output/`

**NOTE:** In UNIX, when you specify the absolute file or folder path, ensure that you use the forward slash ('/'), and not the backward slash ('\').

## Delete operation

Deletes a file from the AEM Forms Server file system or other file system that the server can access.

For information about the General and Route Evaluation property groups, see Commonoperation properties.

### Input properties

Properties for specifying the file to delete.

*Filename*

A `string` value that represents the name of the file you want to delete. Filename includes the absolute path to the file as well as the name of the file.

**Output properties**

Properties that specify whether the file was deleted.

*Success*

(Optional) The location to store the deletion results. The data type is `boolean`. A value of `true` indicates the file was deleted. A value of `false` indicates that the file was not successfully deleted.

**Exceptions**

The exception event attached to this operation can receive FileUtilsException exceptions.

## Exists operation

Discovers the existence of a file on the AEM Forms Server file system.

For information about the General and Route Evaluation property groups, see Commonoperation properties.

**Input properties**

Properties for specifying the file in question.

*Filename*

A `string` value that represents the name of the file that you want to know exists or not. Filename includes the absolute path to the file as well as the name of the file.

**Output properties**

Properties for specifying where to store the operation results.

*Result*

(Optional) The location to store the results of the operation. The data type is `boolean`. A value of `True` indicates the file exists, and `False` indicates the file does not exist.

**Exceptions**

The exception event attached to this operation can receive FileUtilsException exceptions.

## Find operation

Searches for files and directories in the file system of the AEM Forms Server.

For information about the General and Route Evaluation property groups, see Commonoperation properties.

### Input properties

Properties for specifying search parameters.

*Directory*

A `string` value that represents the name of the directory in which to search. The scope of the search includes the directory that you specify (the scope does not include subdirectories). You need to provide the absolute path to the directory.

*Regular Expression*

A `string` value that represents the regular expression used to match the file and directory names. The default is **.*** that matches any file and directory name. For example, if you search for abc.*, the search results will include files and directories that contain abc in their name. For more information about using regular expressions, see Regularexpressions syntax.

*Match Directories*

A `boolean` value that indicates whether to find matching directories. A value of `True` causes directory matches to be returned, and `False` causes no search for directories to occur. The default is `True`.

To specify a literal value of `True` for this property, select Match Directories. To specify a literal value of `False`, deselect Match Directories.

*Match Files*

A `boolean` value that indicates whether to find matching files. A value of `True` causes file matches to be returned, and `False` causes no search for files to occur. The default is `True`.

To specify a literal value of `True` for this property, select Match Files. To specify a literal value of `False`, deselect Match Files.

### Output properties

Properties for specifying where to save results.

*Results*

(Optional) The location to save the results of the search. The data type is `list`. The `list` value contains `string` values that represent the directory and file names.

For information about retrieving values from a list, see Accessingdata in data collections.

### Exceptions

The exception event attached to this operation can receive FileUtilsException exceptions.

## Is Absolute operation

Determines whether a path name is an absolute path.

For information about the General and Route Evaluation property groups, see Commonoperation properties.

### Input properties

Properties for specifying the path.

*Pathname*

A `string` value that represents the path name.

### Output properties

Properties for specifying where to save results.

*Result*

(Optional) The location to store the result. The data type is `boolean`. A value of `True` indicates that the path is absolute and a value of `False` indicates that the path is not absolute.

### Exceptions

The exception event attached to this operation can receive FileUtilsException exceptions.

## Is Directory operation

Determines whether a path name represents a directory.

For information about the General and Route Evaluation property groups, see Commonoperation properties.

### Input properties

Properties for specifying the path.

*Pathname*

A `string` value that represents the path.

### Output properties

Properties for specifying where to save results.

*Result*

(Optional) The location to store the result. The data type is `boolean`. A value of `True` indicates that the path represents a directory and a value of `False` indicates that the path is not that of a directory.

### Exceptions

The exception event attached to this operation can receive FileUtilsException exceptions.

## Is File operation

Determines whether a path represents a file.

For information about the General and Route Evaluation property groups, see Commonoperation properties.

### Input properties

Properties for specifying the path.

*Pathname*

A `string` value represents the path.

### Output properties

Properties for specifying where to save results.

*Result*

(Optional) The location to store the result. The data type is `boolean`. A value of `True` indicates that the path represents a file and a value of `False` indicates that the path is not that of a file.

### Exceptions

The exception event attached to this operation can receive FileUtilsException exceptions.

## Is Hidden operation

Determines whether a file path represents a file that is hidden. The results are determined by the operating system that the AEM Forms Server runs on.

- On UNIX operating systems, the paths of hidden files begin with a period (“.”), for example, `/tmp/.file.txt`.
- On Windows operating systems, the Hidden attribute of files determines whether they are hidden or not.

**NOTE:** If the file does not exist, the operation does not return an error. You should use the is File operation to determine if the file exists before using this operation. (See Is File.)

For information about the General and Route Evaluation property groups, see Commonoperation properties.

### Input properties

Properties for specifying the file path.

*Pathname*

A `string` value that represents the file path.

**Output properties**

Properties for specifying where to save results.

*Result*

(Optional) The location to store the result. The data type is `boolean`. A value of `true` indicates that the path represents a hidden file and a value of `false` indicates that the path is not that of a hidden file.

If the file does not exist, the operating system determines whether `true` or `false` is returned.

**Exceptions**

The exception event attached to this operation can receive FileUtilsException exceptions.

## Make Directory operation

Creates a directory on the file system of the AEM Forms Server.

For information about the General and Route Evaluation property groups, see Commonoperation properties.

**Input properties**

Properties for specifying the directory path.

*Pathname*

A `string` value that represents the fully qualified path of the directory to create. The folder in which the directory is to be created must already exist.

**Output properties**

Properties for specifying where to save results.

*Success*

(Optional) A `boolean` value that indicates whether the directory was successfully created. A value of `true` indicates the directory was created, and `false` indicates the directory was not successfully created.

**Exceptions**

The exception event attached to this operation can receive FileUtilsException exceptions.

## Make Directory Tree operation

Recursively creates a directory on the file system of the AEM Forms Server. If any of the folders in the path of the directory do not exist, they are created.

For information about the General and Route Evaluation property groups, see Commonoperation properties.

### Input properties

Properties for specifying the directory path.

*Pathname*

A `string` value that represents the fully qualified path of the directory to create.

### Output properties

Properties for specifying where to save results.

*Success*

A `boolean` value that indicates whether the directory was successfully created. A value of `true` indicates the directory was created, and `false` indicates the directory was not successfully created.

### Exceptions

The exception event attached to this operation can receive FileUtilsException exceptions.

## Read Document operation

Reads a document from the file system of the AEM Forms Server, and saves the file as a document value in the process data.

For information about the General and Route Evaluation property groups, see Commonoperation properties.

### Input properties

Properties for specifying the file to read.

*Filename*

A `string` value that represents the fully qualified name of the file to read from the file system.

### Output properties

Properties for specifying where to save results.

*Output Document*

(Optional) The location to save the file content. The data type is `document`.

**Exceptions**

The exception event attached to this operation can receive FileUtilsException exceptions.

## Read String operation

Reads a character document from the file system of the AEM Forms Server, and saves the file as a `string` value in the process data.

For information about the General and Route Evaluation property groups, see Commonoperation properties.

**Input properties**

Properties for specifying the file to read.

*Filename*

A `string` value that represents the fully qualified name of the file to read from the file system.

*CharSet*

(Optional) A `string` value that represents the character set to use to read the file. The following values can be specified:

- `Big_Five`
- `GB_2312`
- `ISO_8859_1`
- `ISO_8859_2`
- `ISO_8859_7`
- `KSC_5601`
- `Shift_JIS`
- `US_ASCII`
- `UTF_16`
- `UTF_16BE`
- `UTF_16LE`
- `UTF_8`

**Output properties**

Properties for specifying where to save results.

*Result*

(Optional) The location to save the file content. The data type is `string`.

**Exceptions**

The exception event attached to this operation can receive FileUtilsException exceptions.

## Read XML operation

Reads an XML file from the file system of the AEM Forms Server, and saves the file as an XML document in the process data.

For information about the General and Route Evaluation property groups, see Commonoperation properties.

**Input properties**

Properties for specifying the file to read.

*Filename*

A `string` value that represents the fully qualified name of the file to read from the file system of the AEM Forms Server. Filename can include the following parameters that are replaced with specific values at run time.

| Parameter | Description |
|---|---|
| `%Y` | The current year (four digits) |
| `%y` | The current year (last two digits) |
| `%M` | The current month |
| `%D` | The current day of the month |
| `%d` | The current day of the year |
| `%H` | The current hour using the 24-hour clock |
| `%h` | The current hour using the 12-hour clock |
| `%m` | The current minute |
| `%s` | The current second |
| `%R` | A random number between 0 and 9 |

*CharSet*

(Optional) A `Character Set (FileUtilities)` value that represents the character set that the file uses. The following values are valid:

- `Big_Five`
- `GB_2312`
- `ISO_8859_1`
- `ISO_8859_2`
- `ISO_8859_7`
- `KSC_5601`
- `Shift_JIS`
- `US_ASCII`
- `UTF_16`
- `UTF_16BE`
- `UTF_16LE`
- `UTF_8`

The default is `UTF_8`. If you create a `com.adobe.livecycle.fileutils.CharSet` variable, you must select Find Type in the Type list and search for the `CharSet` type.

**Output properties**

Properties for storing operation results.

*Result*

(Optional) The location in which to save the file content that is read. The data type is `org.w3c.dom.Document`.

**Exceptions**

The exception event attached to this operation can receive FileUtilsException exceptions.

## Rename To operation

Changes the name of a file that resides in the file system of the AEM Forms Server. This operation is useful for moving files to different locations.

For information about the General and Route Evaluation property groups, see Commonoperation properties.

**Input properties**

Properties for specifying the name change for the file.

*Source Filename*

A `string` value that represents the fully qualified name of the file on the file system to rename.

*Destination Filename*

A `string` value that represents the new fully qualified name of the file on the file system.

The source and destination filename can include the following parameters that are replaced with specific values at run time.

| Parameter | Description |
|---|---|
| `%Y` | The current year (four digits) |
| `%y` | The current year (last two digits) |
| `%M` | The current month |
| `%D` | The current day of the month |
| `%d` | The current day of the year |
| `%H` | The current hour using the 24-hour clock |
| `%h` | The current hour using the 12-hour clock |
| `%m` | The current minute |
| `%s` | The current second |
| `%R` | A random number between 0 and 9 |

**Output properties**

Properties for saving the operation results.

*Result*

(Optional) The location to store the new name of the file. The data type is `string`.

**Exceptions**

The exception event attached to this operation can receive FileUtilsException exceptions.

## Set Read Only operation

Sets a file on the file system of the AEM forms Server to read only.

For information about the General and Route Evaluation property groups, see Commonoperation properties.

**Input properties**

Properties for specifying the file.

*Filename*

A `string` value that represents the fully qualified name of the file to set to read only.

**Output properties**

Properties for specifying where to save results.

*Success*

(Optional) A `boolean` value that indicates whether the directory was successfully created. A value of `true` indicates the directory was created, and `false` indicates the directory was not successfully created.

**Exceptions**

The exception event attached to this operation can receive FileUtilsException exceptions.

## Write Document operation

Writes the content of a document that is in the process data to a file in the file system of the AEM Forms Server.

For information about the General and Route Evaluation property groups, see Commonoperation properties.

**Input properties**

Properties that specify the file to write.

*Pathname Pattern*

A `string` value that represents the fully qualified name of the file to create in the file system of the AEM Forms Server. The pathname pattern can include the following parameters that are replaced with specific values at run time.

| Parameter | Description |
|---|---|
| `%Y` | The current year (four digits) |
| `%y` | The current year (last two digits) |
| `%M` | The current month |
| `%D` | The current day of the month |
| `%d` | The current day of the year |
| `%H` | The current hour using the 24-hour clock |
| `%h` | The current hour using the 12-hour clock |

| Parameter | Description |
|---|---|
| `%m` | The current minute |
| `%s` | The current second |
| `%R` | A random number between 0 and 9 |

*Document*

A `document` value that contains the content to write to the file.

If you provide a literal value, clicking the ellipsis button displays the Select Asset dialog box. (See AboutSelect Asset.)

*Make Unique*

A `boolean` value that specifies whether the name of the file is to be made unique. A value of `True` indicates the file name should be made unique and a value of `False` indicates the file name is not unique. The default is `False`.

To specify the literal value of `True`, select Make Unique. To specify the a literal value of `False`, deselect Make Unique.

The following behavior occurs if a file exists with the same file name that is specified for the new file:

- If Make Unique is set to `true`, an index is appended to the name of the file that is created. The index distinguishes the new file from the existing file.
- If Make Unique is set to `false`, the value of the Over Write property determines the behavior:
  - if Over Write is set to `true`, the existing file is overwritten with the new file.
  - If Over Write is set to `false`, an exception is thrown.

*Over Write*

A `boolean` value that specifies whether to replace a file of the same name provided for Pathname Pattern if it exists. A value of `True` indicates the file is replaced and a value of `False` indicates the file name is not replaced. The default is `False`.

To specify the literal value of `True`, select Over Write. To specify the a literal value of `False`, deselect Over Write.

**NOTE:** An exception is thrown when a file exists with the same name that is specified for the new file, Over Write is set to `false`, and Make Unique is set to `false`.

**Output properties**

Properties for saving the operation results.

*Result*

The location to store the name of the file that was written. The data type is `string`.

### Exceptions

The exception event attached to this operation can receive FileUtilsException exceptions.

## Write String operation

Writes the content of A `string` value that is in the process data to a file in the file system of the AEM Forms Server.

For information about the General and Route Evaluation property groups, see Commonoperation properties.

### Input operation

Properties that specify the file to write.

*Pathname Pattern*

A `string` value that represents the fully qualified name of the file to create in the file system of the AEM Forms Server. The pathname pattern can include the following parameters that are replaced with specific values at run time.

| Parameter | Description |
|---|---|
| `%Y` | The current year (four digits) |
| `%y` | The current year (last two digits) |
| `%M` | The current month |
| `%D` | The current day of the month |
| `%d` | The current day of the year |
| `%H` | The current hour using the 24-hour clock |
| `%h` | The current hour using the 12-hour clock |
| `%m` | The current minute |
| `%s` | The current second |
| `%R` | A random number between 0 and 9 |

*Value*

A `string` value that represents the content to write to the file.

*CharSet*

(Optional) A `string` value that represents the character set to use to interpret the text that is written to the file. The following values can be specified:

- `Big_Five`
- `GB_2312`
- `ISO_8859_1`
- `ISO_8859_2`
- `ISO_8859_7`
- `KSC_5601`
- `Shift_JIS`
- `US_ASCII`
- `UTF_16`
- `UTF_16BE`
- `UTF_16LE`
- `UTF_8`

*Make Unique*

A `boolean` value that specifies whether the name of the file is to be made unique. A value of `True` indicates the file name should be made unique and a value of `False` indicates the file name is not unique. The default is `False`.

To specify the literal value of `True`, select Make Unique. To specify the a literal value of `False`, deselect Make Unique.

The following behavior occurs if a file exists with the same file name that is specified for the new file:

- If Make Unique is set to `true`, an index is appended to the name of the file that is created. The index distinguishes the new file from the existing file.
- If Make Unique is set to `false`, the value of the Over Write property determines the behavior:
  - if Over Write is set to `true`, the existing file is overwritten with the new file.
  - If Over Write is set to `false`, an exception is thrown.

*Over Write*

A `boolean` value that specifies whether to replace a file of the same name provided for Pathname Pattern if it exists. A value of `True` indicates the file is replaced and a value of `False` indicates the file name is not replaced. The default is `False`.

To specify the literal value of `True`, select Over Write. To specify the a literal value of `False`, deselect Over Write.

**NOTE:** An exception is thrown when a file exists with the same name that is specified for the new file, Over Write is set to `false`, and Make Unique is set to `false`.

### Output properties

Properties for saving the operation results.

*Result*

(Optional) The location to store the name of the file that was written. The data type is `string`.

**Exceptions**

The exception event attached to this operation can receive FileUtilsException exceptions.

## Write XML operation

Writes an XML document to the file system of the AEM Forms Server.

For information about the General and Route Evaluation property groups, see Commonoperation properties.

**Input properties**

Properties that specify the file to write.

*Pathname Pattern*

A `string` value that represents the fully qualified name of the file to create on the file system of the AEM Forms Server. Filename can include the following parameters that are replaced with specific values at run time.

| Parameter | Description |
|---|---|
| `%Y` | The current year (four digits) |
| `%y` | The current year (last two digits) |
| `%M` | The current month |
| `%D` | The current day of the month |
| `%d` | The current day of the year |
| `%H` | The current hour using the 24-hour clock |
| `%h` | The current hour using the 12-hour clock |
| `%m` | The current minute |
| `%s` | The current second |
| `%R` | A random number between 0 and 9 |

*XML Document*

An `org.w3c.dom.Document` value that represents the content to write to the file.

*CharSet*

(Optional) A `Character Set (FileUtilities)` value that represents the character set that the file uses. The following values are valid:

- `Big_Five`
- `GB_2312`
- `ISO_8859_1`
- `ISO_8859_2`
- `ISO_8859_7`
- `KSC_5601`
- `Shift_JIS`
- `US_ASCII`
- `UTF_16`
- `UTF_16BE`
- `UTF_16LE`
- `UTF_8`

The default is `UTF_8`. If you create a `com.adobe.livecycle.fileutils.CharSet` variable, you must select Find Type in the Type list and search for the `CharSet`type.

*Make Unique*

A `boolean` value that specifies whether the name of the file is to be made unique. A value of `True` indicates that the file name be made unique, and a value of `False` indicates that the file name is not unique. The default is `False`. To specify the literal value of `True`, select Make Unique. To specify the literal value of `False`, deselect Make Unique.

The following behavior occurs if a file exists with the same file name that is specified for the new file:

- If Make Unique is set to `true`, an index is appended to the name of the file that is created. The index distinguishes the new file from the existing file.
- If Make Unique is set to `false`, the value of the Over Write property determines the behavior:
  - if Over Write is set to `true`, the existing file is overwritten with the new file.
  - If Over Write is set to `false`, an exception is thrown.

*Over Write*

A `boolean` value that specifies whether to replace a file of the same name provided for Pathname Pattern if it exists. A value of `True` indicates that the file is replaced, and a value of `False` indicates that the file is not replaced. The default is `False`. To specify the literal value of `True`, select Over Write. To specify the literal value of `False`, deselect Over Write.

**NOTE:** An exception is thrown when a file exists with the same name that is specified for the new file, Over Write is set to `false`, and Make Unique is set to `false`.

### Output properties

Properties for storing operation results.

*Result*

(Optional) A `string` value that represents the actual path of the file that was created.

### Exceptions

The exception event attached to this operation can receive FileUtilsException exceptions.

## Regular expressions syntax

Regular expressions are a series of characters that define a pattern of text. Regular expressions are used in Workbench for expressing search criteria.

Regular expressions are useful when you need to search for text but you do not want to specify the entire content of the text that is returned. Regular expressions are useful in the following situations:

- You only know a portion of the text that you are searching for. For example, if you are searching for files, you may only know the first three letters used to start the file name.
- You need to find multiple instances of text that include a certain pattern. For example, if you want to find all of the files that have the same file extension.

When constructing regular expressions, you need to use the correct syntax:

### Literal characters

The base components of regular expressions are literal characters. For example, the regular expression `abc` matches the sequence `abc`.

*Groups*

Enclosing a sequence of characters in parentheses forms a group, for example `(defg)`. The characters in a group are treated as a single element.

*Any character*

The period (`.`) represents any character. For example `a.bc` matches the sequence of `a` followed by any character followed by `bc`, such as `ambc`.

*The OR operator*

To specify a pattern that includes either one of two characters, you separate them with the pipe character (|). For example, `x|y` matches either the character `x` or `y`. The regular expression `x|(zy)` matches either the character `x` or the sequence `zy`.

### Quantifiers

Quantifiers are special characters used to express a quantity. Quantifiers are applied to the element that directly precedes the quantifier. For example, when a group precedes a quantifier, the quantifier applies to the characters in the group as a whole.

| Character | Quality | Example |
|---|---|---|
| `?` | Zero or one | `a?` matches a sequence of zero or one `a` characters.<br>`abc?` matches a sequence of `ab` followed by zero or one `c` characters, such as `abc` or `ab` |
| `*` | Zero or more | `a*b\|c` matches a sequence of zero or more `a` characters followed by either `b` or `c`, such as `aaaac`. |
| `+` | One or more | `z+` matches a sequence of one or more `z` characters. |
| `{n}` | n times | `efg{3}b` matches a sequence of `ef` followed by three consecutive `g` characters followed by another character, such as `efgggb`. |
| `{n,}` | n or more | `a{3,}` matches a sequence of three or more consecutive `a` characters. |
| `{n,m}` | n or more, but no more than m | `a{3,5}` matches a sequence of three, four, or five consecutive `a` characters. |

**NOTE:** The backslash character (\) is used for escaping special characters so that you can express them as literal characters. For example, \+ matches the plus sign (+). To specify the backslash character (\) in the path on a Windows file system, you use \\ as follows:

```
c:\\file.txt
```

### Character classes

Character classes are a group of characters. You define character classes by enclosing characters and special characters within brackets ([ ]).

| Character | Explanation | Example |
|---|---|---|
| Literals | The characters that you want to include in the character class | `[abc]` specifies `a` or `b` or `c`. |
| ^ | The negation of characters. | `[^abc]` specifies any character except `a`, `b`, and `c`. |
| – | An inclusive range of characters. | `[a-z]`specifies all characters between a and z, including `a` and `z`.<br>`[a-zA-Z]` specifies all characters between `a` and `z`, inclusive (including `a` and `z`), as well as all characters between `A` and `Z`, inclusive. |

| Character | Explanation | Example |
|---|---|---|
| nested brackets | Specifies the union of the character class that is defined by the nested bracket. | `[a-d[m-p]]` specifies all characters between `a` and `d`, inclusive, as well as all characters between `m` and `p`, inclusive. Syntactically equivalent to `[a-dm-p]`. |
| `&&` | Specifies the intersection of characters with a character class. | `[a-z&&[def]]` specifies the intersection of the character range`a-z` with the character class that includes `def`. This example regular expression evaluates to the characters `d`, `e`, or `f`. |

*Predefined character classes*

Several predefined character classes can be expressed using shorthand.

| Character class | Shorthand | Description |
|---|---|---|
| `[0-9]` | `\d` | A digit. |
| `[^0-9]` | `\D` | Any character that is not a digit. |
| `\\x` | `\s` | A white space character. |
| `[^\s]` | `\S` | Any character that is not a white space character. |
| `[a-zA-Z_0-9]` | `\w` | A character that represents a letter of the alphabet |
| `[^\w]` | `\W` | Any character that is not a letter of the alphabet. |

### File Utilities exceptions

The File Utilities service provides the following exceptions for throwing exception events.

#### FileUtilsException

A general exception that is thrown when an error occurs during the execution of a File Utilities service operation.

## 22.26. Form Augmenter

**NOTE:** Effective March 10, 2012, Adobe is deprecating the Guides capabilities of Adobe® AEM Forms® ES. The Guides functionality is available for upgrade purposes only and will be removed from the product after two major releases.

Provides operations that are useful for integrating forms into processes, and enabling them for use with Workspace. For example, Form Augmenter service operations are useful in custom render and submit services for Form, Document Form, or xfaForm variables.

In general, you would call Form Augmenter service operations in the following order:

1) Retrieve the form as a `document` value, for example, a PDF file.
2) Use the InsertWorkflow XFO Data operation or Lookupand Insert Workflow XFO Data operation to insert values into the form.
3) Merge data with the form by using the importData operation.
4) Enable the form for Workspace by using the InjectForm Bridge operation.

For more information about using the Form Augmenter service, see Services Referencefor AEM Forms

.

## Get value of a form field operation

Retrieves a value from a field on a form. The PDF form must be created within Designer 8.2 or later, or Adobe Acrobat 7.0.5 or later.

For information about the General and Route Evaluation property groups, see Commonoperation properties.

### Input properties

Properties to specify the form and form field.

*Input Form Data Document*

A `document` value that represents the XDP data that is merged with the form.

If you provide a literal value, clicking the ellipsis button opens the Select Asset dialog box. (See About-Select Asset.)

*Name Of Form Field*

A `string` value that represents the name of the field on a form. For example, in a form design, in a field named mortgageAmount, you would type `mortgageAmount`.

### Output properties

Property to store the value from a form field.

*Value Of Form Field*

The location to save the value from a form field. The data type is `string`.

### Exceptions

This operation can throw a Java.lang.Exception exception.

## Inject Form Bridge operation

Adds JavaScript code to enable Flex applications and AIR applications to communicate with the PDF form. The service operation enables:

- Adobe PDF form or Adobe Acrobat form to function with Workspace.
- Adobe PDF for Adobe Acrobat form to function with Guides.
- Adobe PDF form to be used with the Document Portfolio Managementbuilding block from the Unified Workspace Solution

If your process uses Adobe XML forms (XDP files), you can render the form to PDF and then use the Inject Form Bridge operation. To render to PDF, you use the renderPDFForm operation that the Forms service provides. For more information on writing JavaScript in Designer, see Preparing a form for use in Workspace in Designer Help.

The functioning of the operation depending on the form:

- If the form already has a Form Bridge XFO component, JavaScript would not be added to the form.
- If the form is certified, the certification will break. The form has to be certified only after adding the Form Bridge XFO component.
- If the form is rights enabled, the encryption will not function. The form has to be rights enabled only after adding the Form Bridge XFO component.
- If the form is digitally signed, the signature will be broken. The form has to be digitally signed only after adding the Form Bridge XFO component.

Workspace provides a Complete button that users click to submit their forms. However, forms can also include submit buttons. When the Inject Form Bridge operation is used on a form, Workspace hides the submit button.

| Form design | Result |
|---|---|
| The form includes no submit button. | Workspace disables the Complete button and users cannot submit the form. The form can be submitted without the Submit button by enabling the Reader Submit option in the Assign Task and Assign Multiple Tasks service operations. See Assign Taskoperation and AssignMultiple Tasks operationfor more information. |
| The form includes one submit button. | Workspace hides the submit button and enables the Workspace Complete button. |
| The form includes a button (indirect submit) that points to a submit button (direct submit)<br>Indirect-submit buttons always take precedence over direct-submit buttons, even if multiple submit buttons exist. Workspace always shows the indirect submit buttons. | Workspace hides the submit button and enables the Workspace Complete button. |

| Form design | Result |
|---|---|
| The form includes multiple indirect-submit buttons that point to one or more direct-submit buttons. | Workspace disables the Workspace Complete button. The user must click the appropriate button on the form to submit it.<br>The user can still save a draft version of the form or take the form offline |
| The form includes either an indirect- or direct-submit button in a repeating subform. | Workspace excludes these buttons for submitting the form in Workspace. |

**NOTE:** When the Submit button that was added to the form with the Process Fields form object is hidden, the button still provides the functionality for submitting the form.

Submit requests are handled by Workspace, which acts as an intermediary between the AEM Forms Server and the form which can be used both offline and online. To understand more about designing your form for Workspace, see Preparing a form for use in Workspace in Designer Help.

### Input properties

Properties to enable the PDF form to be used in Workspace.

*Input PDF Document*

A `document` value that specifies the PDF form to which the Form Bridge will be injected.

If you provide a literal value, clicking the ellipsis button opens the Select Asset dialog box. (See AboutSelect Asset.)

### Output properties

Properties to specify the location to store the PDF form.

*Augmented PDF Document*

The location to save the augmented PDF document. The data type is `document`.

### Exceptions

This operation can throw a Java.lang.Exception exception.

## Insert Workflow XFO Data operation

Adds data fields to the form data, which enables a PDF form created in Designer to be used offline in Workspace. You cannot specify an Acrobat form.

For information about the General and Route Evaluation property groups, see Commonoperation properties.

### Input properties

Properties to specify the form data as an XDP (XML format) data file.

*Input Form Data Document*

An `xml` value that refers to the form data. This is formatted as an XDP (XML format) data file.

If you provide a literal value, clicking the ellipsis button opens the Select Asset dialog box. (See AboutSelect Asset.)

*Submit Target Url*

A `string` value that identifies the location of the Workspace servlet used to handle submit requests for a form.

If you provide a literal value, type the location server name of the AEM Forms Server. For example, to access Workspace, type `http://`*[server name]*`:`*[port]*`/ /rest/services/ProcessManagementDocumentHandlingService/submit` where *[server name]* specifies the name of the AEM Forms Server and *[port]* specifies the port number. The following default port numbers are used by AEM Forms for each application server:

- **JBoss:**`8080`
- **WebLogic:**`7001`
- **WebSphere:**`9080`

  **NOTE:** You can also use HTTP over Secure Socket Layer (SSL) for the Submit Target Url property.

*Task Id*

A `string` value that represents the unique identifier of the task.

If you provide a literal value, type the unique identifier representing the task.

*Return Server Email Address*

A `string` value that represents the email address used by the AEM Forms Server.

If you provide a literal value, type the email address used by the AEM Forms Server. For example, you can type `myadministrator@example.com`.

*Task Status*

A `string` value that represents the status of the assigned task.

If you provide a literal value, type the status of the task using one of the following values.

**1:**

Task was created.

**2:**

Task was created and saved.

**3:**

Task was assigned.

**4:**

Task was assigned and saved.

**100:**

Task was completed.

**101:**

Task deadline has passed.

**102:**

Task was terminated.

*List of Routes*

A `string` value represents a list of routes that the user can choose from. A route represents the actions that occur before the task is considered complete.

If you provide a literal value, type the list of routes, delimited by commas, that are to be selected by the user. For example, you could type `Accept, Deny, Cancel`.

*Assigned User Id*

A `string` value that represents the Global Unique Identifier (GUID) of the user who is assigned the task. You would typically use this to save the GUID of the user who completed the task.

If you provide a literal value, type the GUID that represents the user.

**Output properties**

Properties to specify the location to store the modified form data.

*Modified Data Document*

The location to save the modified form. The data type is `xml`.

**Exceptions**

This operation can throw a Java.lang.Exception exception.

## Lookup and Insert Workflow XFO Data operation

Enables a PDF form created in Designer to be used offline in Workspace. You cannot specify an Acrobat form. The user profile assigned to submit the form, the list of available routes, and the task status are determined by looking up the information using the input Task Id value.

For information about the General and Route Evaluation property groups, see Commonoperation properties.

**Input properties**

Properties to specify the form data as an Acrobat form or an Adobe XML form.

*Input Form Data Document*

An `xml` value that refers to the form data. This is formatted as an XDP data file.

If you provide a literal value, clicking the ellipsis button opens the Select Asset dialog box. (See AboutSelect Asset.)

*Submit Target Url*

A `string` value that represents the location of the Workspace servlet used to handle submit requests for a form.

If you provide a literal value, type the location server name of the AEM Forms Server. For example, to access Workspace, type `http://`*[server name]*`:`*[port]*`/rest/services/ProcessManagementDocumentHandlingService/submit` where *server name* specifies the name of the AEM Forms Server, and *[port]* specifies the port. The following default port numbers are used by AEM Forms for each application server.

- **JBoss:**`8080`
- **WebLogic:**`7001`
- **WebSphere:**`9080`

  **NOTE:** You can also use HTTP over Secure Socket Layer for the Submit Target Url property.

*Task Id*

A `string` value that represents the unique identifier of the task.

If you provide a literal value, type the unique identifier representing the task.

*Return Server Email Address*

A `string` value that represents the email address used by the AEM Forms Server.

If you provide a literal value, type the email address used by the AEM Forms Server. For example, you can type `myadministrator@sampleOrganization.com`.

**Output properties**

Properties to specify the location to store the modified form data document.

*Modified Data Document*

The location to store the modified form data. The data type is `xml`.

### Exceptions

This operation can throw a Java.lang.Exception exception.

## Remove Data Fields operation

Removes a set of data fields from the form data.

For information about the General and Route Evaluation property groups, see Commonoperation properties.

### Input properties

Properties to specify the form data and fields to remove.

*Input Data Document*

An `xml` value that represents the form data.

If you provide a literal value, clicking the ellipsis button opens the Select Asset dialog box. (See AboutSelect Asset.)

*Input Field Map*

A `map` value of sub-type `string` that specifies the data fields to remove from the form data.

### Output properties

Properties to specify the location to store the modified form data document.

*Modified Data Document*

The location to save a form data. The data type is `xml`.

### Exceptions

This operation can throw a Java.lang.Exception exception.

## Form Augmenter exceptions

The Form Augmenter service provides the following exception for throwing exception events.

### Java.lang.Exception

Thrown when an error occurs when augmenting a form.

# 22.27. Form Data Integration

Merges form data into a PDF form. You can import form data into a PDF form and export form data from a PDF form.

In addition, you can import and export interactive PDF form data. An interactive PDF form contains one or more fields for collecting information from a user or displaying custom information. Here are the types of PDF forms:

- A PDF form created in Acrobat is a PDF document that contains form fields.
- A PDF form created in Designer is XML-based.

  *The data format that you can use for importing and exporting depends on the type of form on which you are performing the operation. This data can be in one of these formats:*
- An XFDF file, which is an XML version of the Acrobat form data format.
- An XDP file, which is an XML file that contains form field definitions. It may also contain form field data and an embedded PDF file. An XDP file generated by Designer can be used only if it carries an embedded base-64-encoded PDF document.
- An arbitrary XML file that contains a list of name and value pairs.

For information about using the Form Data Integration service, see Services Reference for AEM Forms

.

## Retaining legacy appearance of PDF Forms in AEM Forms

In AEM Forms, PDF forms rendered with the assistance of Forms, Output, or Form Data Integration (FDI) services have the appearance that is similar to PDF forms in Acrobat X. However, if you are upgrading from the previous version of AEM Forms, and would like to retain the older appearance of PDF forms, you can set a service-level configuration option for the Forms, Output, and FDI services.

For the Forms, Output, or FDI services, when the value of this option (Appearance Compatibility Mode) is set to 9, all PDF forms rendered using the service appear in the legacy appearance. Any other value set for this option results in generated PDF forms appearing with the look and feel enabled by AEM Forms.

1) In the Components view, drill down to **FormDataIntegration** > **Active Services** > **FormDataIntegration:1.1**.
2) Right click FormDataIntegration:1.1 and select Edit Service Configuration.
3) On the Configuration Settings dialog, edit the Appearance Compatibility Mode option to set the version of Acrobat whose appearance mode is to be configured.
4) Click OK to complete the setting.

## exportData operation

Exports form data from a PDF document. The following table shows data formats that can be exported.

| Form type | Export format | Exported Data Format |
|---|---|---|
| Acrobat form | Auto | XFDF |
| Acrobat form | XDP | XFDF (XDP is not valid input for an Acrobat form) |
| Acrobat form | XML | XFDF (XDP is not valid input for an Acrobat form) |
| XML form | Auto | XDP |
| XML form | XDP | XDP |
| XML form | XML | XML |

For information about the General and Route Evaluation property groups, see Commonoperation properties.

**Input properties**

*PDF Document*

A `document` value that represents the PDF document from which data is exported.

If you provide a literal value, clicking the ellipsis button opens the Select Asset dialog box. (See About-Select Asset.)

*Data Format*

The type of data to export. Possible settings are:

- **Auto** XFDF data is exported from an Acrobat form PDF. XDP data is exported from a PDF created in Designer.
- **XDP** XDP data is exported from a PDF created in Designer.
- **XmlData** XML data is exported from a PDF created in Designer.

**Output properties**

*Data Extracted*

The location to store the exported data. The data type is `document`.

**Exceptions**

This operation can throw an ExportFormDataException exception.

## importData operation

Imports XFA or XFDF form data into a PDF document. The following table shows which data formats you can use for each PDF document type.

| Form type | Import format | Result |
|---|---|---|
| Acrobat form | XFDF | Successful |
| Acrobat form | XFA | Unsuccessful |
| XML form | XFA | Successful |
| XML form | XFDF | Unsuccessful |

For information about the General and Route Evaluation property groups, see Commonoperation properties.

### Input properties

*PDF Document*

A `document` value that represents the PDF document to which data is to be imported.

If you provide a literal value, clicking the ellipsis button opens the Select Asset dialog box. (See AboutSelect Asset.)

*Input Data*

A `document` value that represents XFA or XFDF data to import.

If you provide a literal value, clicking the ellipsis button opens the Select Asset dialog box. (See AboutSelect Asset.)

### Output properties

*Data Merged PDF*

The location to store the PDF document that contains the imported data. The data type is `document`.

### Exceptions

This operation can throw an ImportFormDataExceptionexception.

## Form Data Integration exceptions

The Form Data Integration service provides the following exception for throwing exception events.

### ImportFormDataException

Thrown when an error occurs merging form data with a document.

**ExportFormDataException**

Thrown when an error occurs exporting form data from a document.

# 22.28. Forms

**NOTE:** Effective March 10, 2012, Adobe is deprecating the Guides capabilities of Adobe® AEM Forms® ES. The Guides functionality is available for upgrade purposes only and will be removed from the product after two major releases.

The Forms service renders form designs to multiple types of forms, including PDF forms, HTML forms, and Guides. A user can fill the form and send information back to the server by submitting it. Typically, forms are rendered to client applications, such as web browsers, to collect information from end users. You can also use the Forms service to merge data with a form design to populate field values. For example, you can merge predefined data to provide default values in a form.

**TIP:** To test your forms designs, use the FormsIVS sample. (See AEM Forms Samples

*).*

You create form designs in Designer. The Forms service can use these types of form designs to output PDF forms, HTML forms, or Guides:

- Adobe Dynamic XML Form (*.pdf) (Can render PDF only.)
- Adobe Static PDF Form (*.pdf) (Can render PDF only.)
- Adobe XML Form (*.xdp) (Can render all supported outputs.)

For example, you can use the Forms service to render an Adobe XML form (*.xdp) as an HTML form to display to a user. In another step in your process, you can render the same XML form and data as a PDF form to display to another user.

In Designer, special form objects must be added to the form design to render as an HTML form or PDF form for use with Workspace. For information about how to prepare a form for forms workflow, see Using Designer > Creating Forms for forms workflow in the Designer Help

.

For information about using the Forms service, see Services Referencefor AEM Forms

.

For more information, see Designer Help

.

## Retaining legacy appearance of PDF Forms in AEM Forms

In AEM Forms, PDF forms rendered with the assistance of Forms, Output, or Form Data Integration (FDI) services have the appearance that is similar to PDF forms in Acrobat X. However, if you are upgrading from the previous version of AEM Forms, and would like to retain the older appearance of PDF forms, you can set a service-level configuration option for the Forms, Output, and FDI services.

For the Forms, Output, or FDI services, when the value of this option (Appearance Compatibility Mode) is set to 9, all PDF forms rendered using the service appear in the legacy appearance. Any other value set for this option results in generated PDF forms appearing with the look and feel enabled by AEM Forms.

1) In the Components view, drill down to **Forms** > **Active Services** > **FormsService:1.1**.
2) Right click FormsService1.1 and select Edit Service Configuration.
3) On the Configuration Settings dialog, edit the Appearance Compatibility Mode option to set the version of Acrobat whose appearance mode is to be configured.
4) Click OK to complete the setting.

## processFormSubmission operation

Processes the content from a form submitted by a client to the Forms service. This operation can determine the processing state of the form that was submitted, such as the button clicked by a user. You get one of the following results from completing this operation:

- The same content type that was submitted to the Forms service if the form did not complete processing
- XML data or PDF data

  *If the content type of the form passed to this operation is not supported, no processing occurs and the data is returned. Use the processFormSubmission operation to retrieve and process the data from a submitted form.*

  *For example, your application must determine whether the user clicked the submit button and extract the data to save to a process variable from the submitted form. Use the processFormSubmission operation to determine whether the user clicked the submit button by retrieving the value from the Action property after the operation completes execution. Also, use this operation to extract the XML data from the Output XML property and save it to a process variable.*

  *For information about the General and Route Evaluation property groups, see Commonoperation properties.*

### Input properties

Properties for specifying form data for submission.

*Submitted Form Data*

A document value that represents the content that is submitted to the Forms service.

If you provide a literal value, clicking the ellipsis button opens the Select Asset dialog box.

*Environment Variables*

Common ones that you can include are Accept-Language, CONTENT_LENGTH, CONTENT_TYPE, HTTP_FS_REQUEST, and SERVER_NAME. Environment variables are useful for communicating information for server-side scripts or CGI scripts on the client application. For example, specify the CONTENT_-TYPE environment variable to specify the types of content that can be submitted.

*User Agent*

A string value that provides information about the client application, such as a web browser. If `USER_AGENT` is specified as a value in Environment Variables property in this operation, the value specified in this property overrides it. The default value is `Mozilla/3.*`.

If you provide a literal value, select one of these values:

- **Use Server Default: (Default) Use Mozilla/3.***. The User Agent setting is not configurable on the server.
- **Internet Explorer 6.0**: The target application is Microsoft Internet Explorer 6.0.
- **Mozilla Firefox 2.0**: The target application is Mozilla Firefox 2.0.
- **Mozilla Firefox 3.0**: The target application is Mozilla Firefox 3.0.
- **Use Custom Value**: After selecting this option, type the client application on the target device. For example, type `Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1)` for Microsoft Internet Explorer 7.0.

**Form Submission Options properties**

Property for configuring options for submitting a form.

*Submission options*

No Styles: Use any CSS an inline or internal styling.

**Output properties**

Properties for storing the form data submission result.

*Process Form Submission Result Document*

The location in the process data model to store the results of processing a form submission. The data type is document.

**Additional Output properties**

Properties for storing additional information from processing the form submission.

*Clicked Button*

The location in the process data model to store the last button that the user clicked. The value is XFA Scripting Object Model (SOM) expression. The data type is document.

*Output XML*

The location in the process data model to store the XML or HTML data. The result is stored as well-formed XML that can be retrieved. The data type is document.

*Validation Errors List*

The location in the process data model to store a list of validation errors that is retrieved as UTF8-encoded XML. The data type is document.

*Action*

The location in the process data model to store the processing state associated with the form submission. The data type is short. One of these values is returned:

- **0**: The form is in a Submit state, and validated XML data is ready to be processed.
- **1**: The form is in a Calculate state, and calculation results must be written to the web browser.
- **2**: The form is in a Validate state, and calculations and validations must be written to the web browser.
- **3**: The form is in a Next state, and the current page has changed with results that must be written to the web browser. Valid only for HTML forms.
- **4**: The form is in a Previous state, and the current page has changed with results that must be written to the web browser. Valid only for HTML forms.

*Attachments*

The location in the process data model to store the list of attachments that are posted with the form submission. The data type is list of document values.

For information about retrieving values from a list, see Accessingdata in data collections.

*Forms Result*

The location in the process data model to store all the output results in a combined complex value. The data type is FormsResult.

**Exceptions properties**

This operation throws a ProcessFormSubmissionExceptionexception when an error occurs processing a form submit request.

## (Deprecated) renderHTMLForm operation

Retrieves the specified form, merges form data, and transforms it to an HTML form for a client application, such as a web browser. You can only render XDP files (*.xdp) using this operation. You can select the type of HTML form that is created to support different formats. You can also format the appearance of the rendered HTML form by adding a custom CSS file. In addition to using CSS files to format the appearance, consider using layout design techniques to overcome offset shift issues in the rendered HTML form. (See *Using Designer > Creating Forms > Creating HTML forms > Layout considerations for HTML forms* in Designer Help

.) Use this operation to use a form design or form data that are part of an application and you want use the form design and form data as document values.

The benefits of using this operation are as follows:

- Maximizes application portability because all assets are contained in a single application.
- Simplifies the process design because the form design or form data can be assessed as a `document` variable.

  *For example, your application must display HTML forms and permit users to submit the form from any HTML enabled device. HTML forms are also useful because not all devices have Acrobat or Adobe Reader installed on them. Use the (Deprecated) renderHTMLForm operation to render an HTML form from a form design provided as a `document` value. Display the rendered HTML form to a user so that they can submit the form as part of an automated process.*

  **NOTE:** Use the renderHTMLForm(deprecated) operation only when you cannot create a `document` variable to store the form design or form data. You cannot create document variables for form designs or form data that are dynamically referenced on a network, local file system, or HTTP location.

  *For information about the General and Route Evaluation property groups, see Commonoperation properties.*

**Input properties**

The properties that specify the form design and form data for rendering an HTML form.

*Form*

A document value that specifies the form design in an application to render to an HTML form.

If you provide a literal value, clicking the ellipsis button opens the Select Asset dialog box. (See About-Select Asset.)

*Form Data*

A document value that represents the data to merge with the form during rendering. Form data that is provided as XML must be deserialized by using the `deserialize` function. (See deserialize in the *AEM Forms Workbench 9.5 Help* .)

If you provide a literal value, clicking the ellipsis button opens the Select Asset dialog box. (See About-Select Asset.)

**Template Options properties**

The properties that specify form design options, such as its location of the form design, locale, and encoding.

*Content Root URI*

A string value that specifies the URI or an absolute reference, which specifies the folder in the repository to retrieve relative assets used by the form design. For example, if the form design references an image relatively, such as ../myImage.gif, myImage.gif must be located at repository://. The default value is repository://, which points to the root level of the repository.

You can use the following sources for a URI or absolute reference:

- Repository: The repository contains assets that you upload to the AEM Forms Server. The value `repository:///` references the root of the repository. The first two forward slashes are part of the protocol (repository://) and the third forward slash represents the root of the repository.
- Directory in the file system of the AEM Forms Server: You can specify a location on the AEM Forms Server, such as C:*[folder name]*. Using a location on the server is not recommended if you want to ensure portability of your application.
- Network directory: You can specify a location on the network, such as *[folder name]*.
- Web location that is accessible by using HTTP: After you upload a file to location on a web server, you can specify the location using a URL, such as http://*[server name]*:*[port number]*/*[folder name]*.

*Locale*

A string value that specifies the language used to send validation messages to client applications, such as web browsers, when an HTML form is rendered.

If you specify a literal value, select a language from the list or select one of these values:

Use Server Default: (Default) Use the Locale setting that is configured on the AEM Forms Server. The Locale setting is configured using administration console. (See Forms administrationhelp

.)

Use Custom Value: After selecting this option, type the Locale ID of the locale code that is not in the list. For a list of supported locale codes, see http://java.sun.com/j2se/1.5.0/docs/guide/intl/locale.doc.html.

*Character Set*

A string value that specifies the character set used to encode the rendered form.

If you specify a literal value, select the character set to use or one of these values:

Use Server Default: (Default) Select to use the Character Set setting that is configured on the AEM Forms Server. The Character Set setting is configured using Administration Console. (See Forms administrationhelp

.)

Use Custom Value: Select and type the canonical name (Java.nio API) of the encoding set that is not in the list. For a list of character sets, see http://java.sun.com/j2se/1.5.0/docs/guide/intl/encoding.doc.html.

*HTML Options*

Properties to configure parameters for rendering the HTML form.

*Transform To*

A TransformTo value specifies the type of HTML form to generate.

If you provide a literal value, no default is provided. Select one of these values:

AUTO: The Forms service determines the optimal transformation to perform.

XHTML: Generates HTML that is XHTML standard-compliant.

NoScriptXHTML: Generates HTML that is compatible with the CSS2 specification and compliant with XHTML 1.0. The HTML output does not contain client-side JavaScript but can still include server-side JavaScript.

HTML4: Generates HTML that is compatible with older browsers that do not support absolute positioning of HTML elements.

MSDHTML: Generates HTML that is compatible with dynamic HTML for Internet Explorer 5.0 or later.

AHTML: Generates HTML that is compatible with accessibility-enhanced browsers (Internet Explorer 5 or later).

StaticHTML: Generates HTML that does not allow users to enter data in the fields of HTML forms. Static HTML is useful in processes that are for displaying information.

AccessibleXHTML: Generates HTML that is XHTML 1.0 standard-complaint. Elements in the HTML form are positioned by using the relative units `em`. Using relative units helps to ensure that an HTML form appears correctly when tools are used to make content accessible to users with visual impairment.

NoScriptAccessibleXHTML: Generates HTML that is compliant with the XHTML 1.0 standard and CSS2 specification. The XHTML also does not contain client-side JavaScript, although server-side JavaScript can be included. Elements in the HTML form are positioned by using the `em` tag, which is a relative unit. Using relative units helps to ensure that an HTML form appears correctly when tools are used to make content accessible to users with visual impairment.

*HTML Output Type*

An OutputType value that specifies whether to wrap the rendered page with `HTML` or `BODY` tags. Render the HTML with `BODY`tags if you want the rendered HTML page to be part of another HTML page. Select one of these values:

Use Server Default: (Default) Use the default Output type setting that is configured on the AEM Forms Server. The Output type setting is configured using Forms in administration console. (See Forms administrationhelp

.)

Full HTML tags: The rendered page is wrapped with `HTML` tags, which makes it a complete HTML page.

Body Tags: The rendered page is wrapped with `BODY` tags.

*Start Page Number*

An int value that specifies the first page number to start to render in a multi-page HTML form. Page numbering is zero-based, which means that the first page is zero. If no value is provided, zero is used.

*Populate XML Data*

A boolean value that specifies whether the XML data is produced from the form based on the form's current processing state. A value of True means to produce XML data. A value of False means to not produce XML data.

If you provide a literal value, the Populate XML Data check box is deselected by default. Select the check box to generate XML data based on the form's current processing state. Deselect the check box to produce XML data that is not based on the processing state of the form.

*User Agent*

A string value that provides information about the client application, such as a web browser. If `USER_AGENT` is specified as a value in Environment Variables property in this operation, the value specified in this property overrides it. The default value is `Mozilla/3.*`.

If you provide a literal value, select one of these values:

Use Server Default: (Default) Use Mozilla/3.*. The User Agent setting is not configurable on the server.

Internet Explorer 6.0: The target application is Microsoft Internet Explorer 6.0.

Mozilla Firefox 2.0: The target application is Mozilla Firefox 2.0.

Mozilla Firefox 3.0: The target application is Mozilla Firefox 3.0.

Use Custom Value: After selecting this option, type the client application on the target device. For example, type `Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1)` for Microsoft Internet Explorer 7.0.

**Advanced HTML Options properties**

Properties for configuring parameters that control the HTML appearance.

*Digital Signature CSS URI*

A string value that specifies the URI of the CSS file, such as `C:\\customds.css`, for a custom style sheet to use for digital signature user interfaces in HTML forms. The AEM Forms Server must have access to the location of the URI.

**NOTE:** To create a custom CSS file, use the FormsIVS sample to generate a custom CSS based on the form design and output type you provide.

*Custom CSS URI*

A string value that specifies the URI, such as `C:\customcss.css`, for a custom cascading style sheet to use instead of the internal one emitted as part of HTML form. Review the custom CSS file whenever changes are made to the form design:

- A new type of form object is added to the form design, such as button, text, text field, or check box.
- A form object is removed from the form design.
- Moving a form object, such as label or field, from a child subform to its parent or another subform.

  *The AEM Forms Server must have access to the location of the URI.*

  **NOTE:** To create a custom CSS file, use the FormsIVS sample to generate a custom CSS based on the form design and output type you provide.

*HTML Toolbar*

An HTMLToolbar value that specifies the type of toolbar that is rendered for HTML forms.

If you provide a literal value, select one of these values:

Disabled: (Default) The HTML toolbar is disabled.

Vertical: The HTML toolbar appears in a vertical position.

Horizontal: The HTML toolbar appears in a horizontal position.

*HTML Toolbar URI*

A string value that specifies an URI location of the HTML toolbar resources to include in the rendered HTML form.

*Generate Tab Index*

A boolean value that specifies whether to create a tab index for each field in the rendered HTML form. When an HTML form with tab indexes is embedded into another HTML page that does not have tab indexes, tabbing inconsistencies can occur. Do not generate tab indexes for each form field when you want to embed the HTML form into another HTML form that does not have tab indexes.

If you provide a literal value, the Generate Tab Index check box is deselected by default. Select the Generate Tab Index check box to create tab indexes for each form field in the HTML form. Deselect the check box when you do not want to create tab indexes for each form field.

*Style Generation Level*

A StyleGenerationLevelvalue that specifies the styling to output for the rendered HTML form. Internal styles refer to spacing and positional CSS styles. Inline styles refer to appearance styles, such as color, font, and background color.

If you provide a literal value, select one of these values:

Inline And Internal Styles: (Default) Use both inline and internal CSS styles. Internal styles are used for spacing and positional CSS styles. Inline styles are used for appearance styles, such as color, font, and background color.

Only Inline Styles: Use only inline CSS styles from the custom CSS file. This option removes all internal styles inside the `head` tag on HTML pages.

No Styles: Do not use any CSS on inline or internal styling. Only styles that are passed by using an external CSS file are used.

**Render Options properties**

Properties for the rendering options for an HTML form.

*Cache Form On Server*

A boolean value that specifies whether the form design is cached on the server. Caching forms on the server improves performance for forms that are rendered on the server. Processing instructions that are embedded into the form design and the Form Rendering Cache Enabled option determines how caching is performed. For information about configuring the Rendering Cache Enabled option in administration console, see Configuring Forms in Administration Help.

If you provide a literal value, the Better Performance For Templates That Don't Change Often check box is selected by default. Select the check box to render a cache to the form design on the AEM Forms Server. Deselect the check box if caching the form is not required.

*Stand Alone Rendition*

A boolean value that specifies whether the form can be rendered without state information. State information is used for rendering interactive forms that require user interaction with the server for submissions. When you render a PDF form with state information, it is rendered with JavaScript, which allows the form to be used offline.

If you provide a literal value, the Render Without State Information check box is selected by default:

- Select the check box to render a PDF form with state information and embedded JavaScript. The JavaScript code runs on the client with no interaction with the server. In addition, the form can be used offline.
- Deselect the check box to render a PDF form without state information and embedded JavaScript. The form is rendered by the Forms service after server-side calculations are performed. The results of the calculation are displayed in the rendered form. Because of the required interaction with the Forms service, the form cannot be used offline.

*Application Web Root*

A string value that specifies the URL that represents the root location that is used to access application-specific web content. This value is combined with the value of the Target URL option to construct an absolute submit URL.

*Target URL*

A string value that specifies the URL to access a web service or Java servlet that receives data from the client application when a user submits the form. Setting a value in this option sets the target URL in the form design to the value specified by this property. If this option is not an absolute URL, it is combined with the value from the Application Web Root option to construct an absolute URL

**NOTE:** The Target URL property must be left empty for form designs created in Designer 8.2 to use the URL that is configured for the Submit button. Form designs created in Designer 9 and later with a Target Version set to Acrobat and Adobe Reader 9.1 or later always use the URL configured in the Submit button.

For Workspace, type the value of `http://`*[server name]*`:`*[port]*`/workspace-server/submit`, where *[server name]* is the name of the server where AEM Forms is deployed, and *[port]* is the port that the application server uses to provide HTTP access for client software. The default ports for the supported application servers are as follows:

- JBoss: 8080
- WebLogic: 7001
- WebSphere: 9080

*Base URL*

A string value that specifies the URL, which is the HTTP-equivalent of the Content Root URI. This value is required only when you render HTML forms that include href references to external dependencies, such as images or scripts. When a dependency path is absolute, this value is ignored.

**Additional Options properties**

Properties for specifying additional options when rendering an HTML form.

*Form Model*

A FormModel value that specifies the location where scripts that are embedded in the form are executed.

If you provide a literal value, select one of these values:

**Use Form Template Default:**

(Default) The Forms service checks the form design to determine whether to render on the client or server.

**Client Side:**

The form is rendered on the client. Do not use scripts that run on the server. When this value is selected, scripts that run on the server generate a warning on the AEM Forms Server.

**Server Side:**

The form is rendered on the server.

**Both Server and Client side:**

The form is rendered on both the server and the client.

*XCI URI*

A string value that specifies the URI location of the XCI file to use for rendering. If the root is not specified, the file is assumed to reside in the location where the AEM Forms EAR files are deployed. For example, in a JBoss turnkey installation, you can see default XCI files in the *[Install folder]*/jboss/server/lc_turnkey/svcdata/XMLFormService folder. *[Install folder]* is the location where the AEM Forms is installed. For information about creating a custom XCI file, see Designer Help

.

*Attachments*

A map of document values that specifies the attachments that are rendered with the form.

**Output properties**

Properties for storing the HTML form.

*Rendered Form*

The location in the process data model to store the results of the rendered form. The data type is document.

**Additional Output properties**

Properties for storing the results of rendering an HTML form.

*Output XML*

The location in the process data model to store the well-formed XML that represents the content generated by the operation. The data type is document.

*Page Count*

The location in the process data model to store the number of pages. The data type is long.

*Locale*

The location in the process data model to store the locale code of the rendered HTML form. The data type is string.

*HTML Output Type*

The location in the process data model to store the generated HTML data that includes only the `BODY` element and its contents. The data type is string.

*Forms Result*

The location in the process data model to store all the output results as a combined complex value. The data type is FormsResult.

**Exceptions**

This operation throws a RenderFormException exception when an error occurs rendering an HTML form.

## renderHTMLForm operation (deprecated)

**NOTE:** This operation is deprecated. Use the (Deprecated)renderHTMLForm operation instead. It is recommended that when you upgrade a process, you change it to use the (Deprecated) renderHTML-Form operation. Only use the (Deprecated) renderHTMLForm operation when you cannot reference a form design or form data using a *`document`* object. (See Aboutdeprecated operations.)

Retrieves the specified form, merges form data, and transforms it to an HTML form for a client application, such as a web browser. You can only render XDP files (*.xdp) using this operation. You can select the type of HTML form that is created to support different formats. You can also format the appearance of the rendered HTML form by providing a custom CSS file. In addition to using CSS files to format the appearance, consider using layout design techniques to overcome offset issues in the rendered HTML

form. (See *Using Designer > Creating Forms > Creating HTML forms > Layout considerations for HTML forms* in Designer Help

.) Use this operation when you want to use form designs or form data from a network, local file system, or HTTP location that change.

For example, your application must display HTML forms and permit users to submit the form from any HTML enabled device. The form design you are rendering is in a network location, which permits you to assess the form design by reference. HTML forms are also useful when you cannot guarantee that Acrobat or Adobe Reader is installed on the client. Use the renderHTMLForm (deprecated) operation to render a form as HTML to the user so that they can submit it as part of an automated process.

For information about the General and Route Evaluation property groups, see Commonoperation properties.

**Input properties**

The properties that specify the form design, form data, and options when rendering an HTML form.

*Form To Render*

A string value that specifies the name of the XDP file. This value is combined with the value of the Content Root URI property in this operation to construct an absolute path to the form.

If you provide a literal value, the name of the form must be typed in the box below the Form To Render property.

For example, a folder in the repository named *form designs* contains the form design named *form.xdp*. To access the form design, type `/form.xdp` in the Form To Render property and `repository:///formdesigns` in the Content Root URI property. The absolute path created is repository:///formdesigns/form.xdp, which is required to access the form design.

*Transform To*

A TransformTo value specifies the type of HTML form to generate.

If you provide a literal value, no default is provided. Select one of these values:

**AUTO:**

The Forms service determines the optimal transformation to perform.

**XHTML:**

Generates HTML that is XHTML standard-compliant.

**NoScriptXHTML:**

Generates HTML that is compatible with the CSS2 specification and compliant with XHTML 1.0. The HTML output does not contain client-side JavaScript but can still include server-side JavaScript.

**HTML4:**

Generates HTML that is compatible with older browsers that do not support absolute positioning of HTML elements.

**MSDHTML:**

Generates HTML that is compatible with dynamic HTML for Internet Explorer 5.0 or later.

**AHTML:**

Generates HTML that is compatible with accessibility-enhanced browsers (Internet Explorer 5 or later).

**StaticHTML:**

Generates HTML that does not allow users to enter data in the fields of HTML forms. Static HTML is useful in processes that are for displaying information.

**AccessibleXHTML:**

Generates HTML that is XHTML 1.0 standard-complaint. Elements in the HTML form are positioned by using the relative units `em`. Using relative units helps to ensure that an HTML form appears correctly when tools are used to make content accessible to users with visual impairment.

**NoScriptAccessibleXHTML:**

Generates HTML that is compliant with the XHTML 1.0 standard and CSS2 specification. The XHTML also does not contain client-side JavaScript, although server-side JavaScript can be included. Elements in the HTML form are positioned by using the `em` tag, which is a relative unit. Using relative units helps to ensure that an HTML form appears correctly when tools are used to make content accessible to users with visual impairment.

*Form Data*

A document value that represents the data to merge with the form during rendering. Form data that is provided as XML must be deserialized using the `deserialize` function. (See deserialize in the *AEM Forms Workbench 9 Help* .)

If you provide a literal value, clicking the ellipsis button opens the Select Asset dialog box. (See About-Select Asset.)

*HTML Render Options*

An HTMLRenderSpec value that represents run-time options for rendering an HTML form.

If you provide a literal value, you can modify the following fields that appear:

**HTML Output Type:**

Sets whether to wrap the rendered page with `HTML` or `BODY` tags. Render the HTML with `BODY`tags if you want the rendered HTML page to be part of another HTML page. Select one of these values:

**Use Server Default:**

(Default) Use the default Output type setting that is configured on the AEM Forms Server. The Output type setting is configured using Administration Console. (See Forms administration help

.)

**Full HTML tags:**

The rendered page is wrapped with `HTML` tags.

**Body Tags:**

The rendered page is wrapped with `BODY` tags.

**Character Set:**

Sets the character set used to encode the output byte stream. Select the character set to use or one of these values:

**Use Server Default:**

(Default) Use the Character Set setting that is configured on the AEM Forms Server. The Character Set setting is configured using administration console. (See Forms administrationhelp

.)

**Use Custom Value:**

After selecting this value, type the canonical name (Java.nio API) of the encoding set that is not in the list. For a list of character sets, see http://java.sun.com/j2se/1.5.0/docs/guide/intl/encoding.doc.html.

**Start Page Number:**

Sets the first page number to start to render in a multi-page HTML form. Page numbering is zero-based, which means that the first page is zero. The default value is `0`.

**Locale:**

Sets the language used to send validation messages to client applications, such as web browsers, when an HTML form is rendered. Select a language from the list or select one of these values:

**Use Server Default:**

(Default) Use the Locale setting that is configured on the AEM Forms Server. The Locale setting is configured using Forms in administration console. (See Forms administrationhelp

.)

**Use Custom Value:**

After selecting this option, type the Locale ID of the locale code that is not in the list. For a list of supported locale codes, see http://java.sun.com/j2se/1.5.0/docs/guide/intl/locale.doc.html.

**Cache Form On Server:**

Sets whether the rendered form is cached on the server to improve performance. Processing instructions that are embedded into the form design and the Form Rendering Cache Enabled option determines how caching is performed. For information about configuring the Rendering Cache Enabled option in administration console. (See Configuring Forms.)

**False:**

The form is not cached on the server.

**True:**

The form is cached on the server.

**Populate XML Data:**

Sets whether the XML data is produced from the form based on the form's current processing state. Select one of the Select one of these values:

**False:**

(Default) Do not produce the XML data.

**True:**

Produce the XML data.

**Stand Alone Rendition:**

Sets whether the form can be rendered without state information. State information is used for rendered forms that require user interaction with the server for submissions. Select one of these values:

**False:**

(Default) The form is rendered without state information and without embedded JavaScript that runs on the client web browser. The Forms service renders the form after server-side calculations are performed and the results are displayed in the rendered form. Because of the required interaction with the Forms service, the form cannot be used when offline.

**True:**

The form is rendered with state information and with embedded JavaScript. The JavaScript code runs on the client with no interaction with the server. In addition, the form can be used offline.

**Form Model:**

Sets the location where scripts embedded into the form are executed. Select one of these values:

**Use Form Template Default:**

(Default) The Forms service checks the form design to determine whether to render on the client or server.

**Client Side:**

The form is rendered on the client. Do not use scripts that run on the server. When this value is selected, scripts that run on the server generate a warning on the AEM Forms Server.

**Server Side:**

The form is rendered on the server.

**Both Server and Client side:**

The form is rendered on both the server and the client.

**XCI URI:**

Sets a value to specify the URI location of the XCI file to use for rendering. If the root is not specified, the file is assumed to reside in the location where the AEM Forms EAR files are deployed. For example, in a JBoss turnkey installation, you can see default XCI files in the *[Install folder]*/jboss/server/lc_turnkey/svcdata/XMLFormService folder. *[Install folder]* is the location where the AEM Forms is installed. For information about creating XCI files, see Designer Help

.

**Digital Signature CSS URI:**

Sets the URI, such as `C:\\customds.css`, for a custom style sheet to use for digital signature user interfaces in HTML forms. The AEM Forms Server must have access to the location of the URI.

**NOTE:** To create a custom CSS file, use the FormsIVS sample to generate a custom CSS based on the form design and output type you provide.

**Custom CSS URI:**

Sets the URI, such as `C:\customcss.css`, for a custom cascading style sheet to use instead of the internal one emitted as part of the HTML form. Review the custom CSS file for changes when changes are made to the form design.

- A new type of form object is added to the form design, such as button, text, text field, or check box.
- A form object is removed from the form design.
- Moving a form object, such as label or field, from a child subform to its parent or another subform.

*The AEM Forms Server must have access to the location of the URI.*

**NOTE:** To create a custom CSS file, use the FormsIVS sample to generate a custom CSS based on the form design and output type you provide.

**HTML Toolbar:**

Sets the type of toolbar that is rendered for HTML forms. Select one of these values:

**Disabled:**

(Default) The HTML toolbar is disabled.

**Vertical:**

The HTML toolbar appears in a vertical position.

**Horizontal:**

The HTML toolbar appears in a horizontal position.

**HTML Toolbar URI:**

Sets the URI location of the HTML toolbar resources to include in the rendered HTML form.

**Generate Tab Index**

Sets whether to create a tab index for each field in the rendered HTML form. When an HTML form with tab indexes is embedded into another HTML page that does not have tab indexes, tabbing inconsistencies can occur. Select one of these values:

**True:**

(Default) Tab indexes are created for each field in the rendered HTML form.

**False:**

Tab indexes are not created for each field in the rendered HTML form. Use this option when you embed the HTML form into another HTML form that does not have tab indexes.

**Style Generation Level:**

Sets the styling to output for the rendered HTML form. Internal styles are used for spacing and positional CSS styles. Inline styles are used for appearance styles, such as color, font, and background color. Select one of these values:

**Inline And Internal Styles:**

(Default) Use both inline and internal CSS styles. Internal styles use spacing and positional CSS styles. Inline styles use appearance styles, such as color, font, and background color.

**Only Inline Styles:**

Use only inline CSS styles from the custom CSS file.

**No Styles:**

Use any CSS an inline or internal styling.

*User Agent*

A string value that provides information about the client application, such as a web browser. If `USER_AGENT` is specified as a value in Environment Variables property in this operation, the value specified in this property overrides it. The default value is `Mozilla/3.*`.

If you provide a literal value, select one of these values:

**Use Server Default:**

(Default) Use Mozilla/3.*. The User Agent setting is not configurable on the server.

**Internet Explorer 6.0:**

The target application is Microsoft Internet Explorer 6.0.

**Mozilla Firefox 2.0:**

The target application is Mozilla Firefox 2.0.

**Mozilla Firefox 3.0:**

The target application is Mozilla Firefox 3.0.

**Use Custom Value:**

After selecting this option, type the client application on the target device. For example, type `Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1)` for Microsoft Internet Explorer 7.0.

*URL Options*

A URLSpec value that specifies the URI run-time information required to render an HTML form.

If you provide a literal value, you can modify the following fields that appear:

**Application Web Root:**

Sets the URL that represents the root location that is used to access application-specific web content. This value is combined with the value of the Target URL option to construct an absolute submit URL.

**Target URL:**

Sets the URL to access a web service or Java servlet that receives data from the client application when a user submits the form. Setting a value in this option sets the target URL in the form design to the value specified by this property. If this option is not an absolute URL, it is combined with the value from the Application Web Root option to construct an absolute URL.

**NOTE:** The Target URL property must be left empty for form designs created in Designer 8.2 to use the URL that is configured for the Submit button. Form designs created in Designer 9 and later with a Target Version set to Acrobat and Adobe Reader 9.1 or later, always use the URL configured in the Submit button.

For Workspace, type the value of `http://`*[server name]*`:`*[port]*`/workspace-server/submit`, where *[server name]* is the name of the server where AEM Forms is deployed, and *[port]* is the port that the application server uses to provide HTTP access for client software. The following ports are the default ports for the supported application servers.

- JBoss: 8080
- WebLogic: 7001

- WebSphere: 9080

**Content Root URI:**

Sets the URI or an absolute reference, which specifies the location in the repository to retrieve a form design. This value is combined with the value of the Form To Render property in this operation to construct an absolute path to the form.

You can use the following sources for a URI or absolute reference:

- Repository: The repository contains assets that you upload to the AEM Forms Server. The value `repository:///` references the root of the repository. The first two forward slashes are part of the protocol (repository://) and the third forward slash represents the root of the repository. For example, the documents folder is created below the root of the repository.
- Directory in the file system of the AEM Forms Server: You can specify a location on the AEM Forms Server, such as C:*[folder name]*. Using a location on the server is not recommended if you want to ensure portability of your application.
- Network directory: You can specify a location on the network such as \\[*folder name.]*
- Web location that is accessible by using HTTP: After uploading a file to a web server, you can specify the location by using a URL, such as http://*[server name]*:*[port number]*/*[folder name]*.

  *For example, a folder in the repository named form designs contains the form design named form.xdp. To access the form design, type `/form.xdp` in the Form To Render property and `repository:///formdesigns` in the Content Root URI property. The absolute path that is created is repository:///formdesigns/form.xdp, which is required to access the form design.*

**Base URL:**

Sets the URL, which is the HTTP-equivalent of the Content Root URI. This value is required only when you render HTML forms (renderHTMLForm (deprecated) and renderHTMLForm operations) that include HREF references to external dependencies, such as images or scripts. When a dependency path is absolute, this value is ignored.

*Attachments*

A map of document values that specifies the attachments that are rendered with the form.

**Output properties**

Properties for storing the HTML form and the results of rendering the HTML form.

*Rendered Form*

The location in the process data model to store the results of the rendered form. The data type is document.

*Output XML*

The location in the process data model to store the well-formed XML that represents the content generated by the operation. The data type is document.

*Page Count*

The location in the process data model to store the number of pages. The data type is long.

*Locale*

The location in the process data model to store the locale code of the rendered HTML form. The data type is string.

*HTML Output Type*

The location in the process data model to store the generated HTML data that includes only the `BODY` element and its contents. The data type is string.

*Forms Result*

The location in the process data model to store all the output results as a combined complex value. The data type is FormsResult.

**Exceptions**

This operation throws a RenderFormException exception when an error occurs rendering an HTML form.

## renderPDFForm operation

Retrieves the specified form, merges form data, and transforms it to a PDF form for a client application, such as a Adobe Reader. Use this operation when you use a form design or form data that is saved in an application and used as `document` values in the process.

The benefits of using this operation instead of the renderPDFForm (deprecated) operation are as follows:

- Maximizes application portability because all assets are contained in a single application.
- Simplifies the form process design because the form design can be assessed using `document` variable.

  *For example, your application must render a PDF form that is packaged as part of your application, instead of directly. The user that is displayed the PDF form must digitally sign it in Acrobat. A PDF form has embedded security features and the ability for users to digitally sign the form. Use the renderPDFForm operation to render the form design to an interactive PDF form from the application. The PDF form permits users to sign the form and submit it as a part of an automated process.*

  **NOTE:** Use the renderPDFForm operation when you cannot create a `document` variable to store the form design or form data. You cannot create document variables for form designs or form data that are dynamically referenced on a network, local file system, or HTTP location.

  *For information about the General and Route Evaluation property groups, see Commonoperation properties.*

**Input properties**

The properties that specify the form design, form data, and options when rendering a PDF form.

*Form*

A document value that specifies the form design in an application to render.

If you provide a literal value, clicking the ellipsis button opens the Select Asset dialog box. (See About-Select Asset.)

*Form Data*

A document value that represents the data to merge with the form during rendering. Form data that is provided as XML must be deserialized by using the `deserialize` function. (See deserialize in the *AEM Forms Workbench 9.5 Help* .)

If you provide a literal value, clicking the ellipsis button opens the Select Asset dialog box. (See About-Select Asset.)

**Template options properties**

The properties that specify configuration options for the form design.

*Content Root URI*

A string value that specifies the URI or an absolute reference, which specifies the folder in the repository to retrieve relative assets used by the form design, such as images. For example, if the form design references an image relatively, such as ../myImage.gif, myImage.gif must be located at repository://. The default value is repository://, which points to the root level of the repository.

You can use the following sources for a URI or absolute reference:

- Repository: The repository contains assets that you upload to the AEM Forms Server. The value `repository:///` references the root of the repository. The first two forward slashes are part of the protocol (repository://) and the third forward slash represents the root of the repository.
- Directory in the file system of the AEM Forms Server: You can specify a location on the AEM Forms Server, such as C:*[folder name]*. Using a location on the server is not recommended if you want to ensure portability of your application.
- Network directory: You can specify a location on the network such as *[folder name]*.
- Web location that is accessible by using HTTP: After you upload a file to a web server, you can specify the location by using a URL, such as http://*[server name]*:*[port number]*/*[folder name]*.

  *If you provide a literal value, type a value to specify the URI or absolute reference.*

*Locale*

A string value that specifies the language used to send validation messages to client applications, such as web browsers, when an HTML form is rendered.

If you provide a literal value, select a language from the list or select one of these values:

Use Server Default:

(Default) Use the Locale setting that is configured on the AEM Forms Server. The Locale setting is configured using administration console. (See Forms administrationhelp

.)

Use Custom Value:

Use a locale that is not available in the list. After selecting this value, in the box beside the list, type the Locale ID of the locale to use. For a list of supported locale codes, see http://java.sun.com/j2se/1.5.0/docs/guide/intl/locale.doc.html.

*Character Set*

A string value that specifies the character set used to encode the output byte stream.

If you provide a literal value, select a character set from the list or one of these values.:

Use Server Default:

(Default) Use the Character Set setting that is configured on the AEM Forms Server. The Character Set setting is configured using Administration Console. (See Forms administrationhelp

.)

Use Custom Value:

Use a character that is not available in the list. After selecting this value, in the box beside the list, type the canonical name (Java.nio API) of the encoding set to use. For a list of character sets, see http://java.sun.com/j2se/1.5.0/docs/guide/intl/encoding.doc.html.

**PDF Options properties**

Properties that specify configuration parameters for the rendered PDF form.

*Acrobat Version*

An acrobatVersion value that specifies the minimum version of Acrobat and Adobe Reader required to open the rendered PDF form.

If you provide a literal value, select one of these values:

Use Form Template Default:

(Default) The Target Version setting in the form design determines minimum version of Acrobat or Adobe Reader. In addition, the form design determines the PDF Version.

Acrobat and Adobe Reader 6 or later:

Acrobat or Adobe Reader 6 and later can open the PDF form.

Acrobat and Adobe Reader 7.0 or later:

Acrobat or Adobe Reader 7 and later can open the PDF form.

Acrobat and Adobe Reader 7.0.5 or later:

Acrobat or Adobe Reader 7.0.5 and later can open the PDF form

Acrobat and Adobe Reader 8 or later:

Acrobat or Adobe Reader 8 and later can open the PDF form

Acrobat and Adobe Reader 8.1 or later:

Acrobat or Adobe Reader 8.1 and later can open the PDF form.

Acrobat and Adobe Reader 9 or later:

Acrobat or Adobe Reader 9 and later can open the PDF form

*Accessible (Tagged) PDF*

A boolean value that specifies whether to create a tagged Adobe PDF form. A tagged PDF form defines a set of standard structure types and attributes that support the extraction of page content and reuse for other purposes. This property is intended for use by tools that perform the following types of operations:

- Simple extraction of text and graphics for pasting into other applications.
- Automatic reflow of text and associated graphics to fit a page of a different size than was assumed for the original layout.
- Processing text for such purposes as searching, indexing, and spell-checking.
- Conversion to other common file formats (such as HTML, XML, and RTF) with document structure and basic styling information preserved.
- Making content accessible by screen reader software.

  *If you provide a literal value, the Accessible (Tagged) PDF check box is selected by default. Select the check box to render an accessible PDF form. Deselect the check box if you do not want to render an accessible PDF.*

*Linearized PDF*

A boolean value that specifies whether to render a linearized PDF form. A linearized PDF form is organized so that it supports incremental access in a network environment. A linearized PDF form is optimized for use with web applications. A non-linearized PDF form is best for use in non-web applications. For example, a linearized PDF can be displayed in a web browser before the entire PDF document is downloaded.

If you provide a literal value, the Create PDF For Quicker Display In A Browser check box is selected by default. Select the check box to render a linearized PDF form. Deselect the check box to generate a non-linearized PDF form.

**Render options properties**

Properties for configuring rendering options for the PDF form.

*Cache Form On Server*

A boolean value that specifies whether the form design is cached on the server. Caching forms on the server improves performance for forms that are rendered on the server. Processing instructions embedded in the form design and the Form Rendering Cache Enabled option determines how caching is performed. For information about configuring the Rendering Cache Enabled option in administration console. (See Configuring Forms.)

If you provide a literal value, the Better Performance For Templates That Don't Change Often check box is selected by default. Select the check box to render a cache to the form design on the server. Deselect the check box if caching the form is not required.

*Client Cache*

A boolean value that specifies whether the rendered PDF form is cached on the client.

If you provide a literal value, the Cache Form In Web Browser Cache check box is deselected by default. Select the check box to cache the rendered PDF form on the client. Deselect the check box if caching is not required.

*Generate Server Appearance*

A boolean value that specifies whether the appearance for the PDF form is generated on the server. Appearances include the layout of fields and graphical elements in the PDF form. When you generate the appearance on the server, the form is rendered on the server and merged with data. Generate the appearance when you want to use the rendered PDF form in a subsequent operation. For example, if you need to apply a digital signature to the rendered PDF form on the server, generate the appearance on the server before applying the digital signature. If the appearance is not generated on the server, it is generated in Acrobat or Adobe Reader when the PDF form is rendered on the client, which invalidates the digital signature that was applied.

If you provide a literal value, the Generate Appearances On Server check box is deselected by default. Select the check box to generate appearances on the server. Deselect the check box to generate the appearance when the PDF form is rendered in Acrobat or Adobe Reader.

*Stand Alone Rendition*

A boolean value that specifies whether the form can be rendered without state information. State information is used for rendering interactive forms that require user interaction with the server for submissions. When you render a PDF form with state information, it is rendered with JavaScript, which allows the form to be used offline.

If you provide a literal value, the Render Without State Information check box is selected by default.

- Select the check box to render a PDF form state information and embedded JavaScript. The JavaScript code runs on the client with no interaction with the server. In addition, the form can be used offline.
- Deselect the check box to render a PDF form without state information and without embedded JavaScript. The Forms service renders the form after server-side calculations are performed and the results are displayed in the rendered form. Because of the required interaction with the Forms service, the form cannot be used offline.

*Render At Client*

A RenderAtClient value that specifies whether to enable the delivery of PDF content by using the client-side rendering capability. Client-side rendering improves the performance of the Forms service. Rendering only occurs for users of Acrobat 7 and Adobe Reader or later. This property applies only to PDF, PDFForm, or PDFMerge transformations.

If you provide a literal value, select one of these values:

**Use Server Default**

(Default) Use the Render At setting version on the server to open PDF forms rendered by the Forms service. The Render At setting is configured in administration console. (See Forms administrationhelp

.)

**Use Form Template Default**

The Forms service determines the form rendition based on the setting in the form design.

**Yes**

A dynamic PDF form is generated and rendering occurs in Acrobat. Rendering occurs only on Acrobat 7.0 or later.

**No**

A static PDF form is generated. Rendering on the client does not occur.

*Target URL*

A string value that specifies the URL to a web service or Java servlet that receives data from the client application. The web service or Java servlet receives the data when the user submits the form. Setting a value in this option sets the target URL in the form design to the value specified by this property. If this option is not an absolute URL, it is combined with the value from the Application Web Root option to construct an absolute URL

For Workspace, type the value of `http://`*[server name]*`:`*[port]*`/workspace-server/submit`, where*[server name]* is the name of the server where AEM Forms is deployed, and *[port]* is the port that the application server uses to provide HTTP access for client software. The default ports for the supported application servers are as follows:

- JBoss: 8080
- WebLogic: 7001
- WebSphere: 9080

*If you provide a literal value, type a value to specify the URL.*

**Additional Options properties**

Properties for using extra rendering options from the Forms service.

*XCI URI*

A string value that sets the URI location of the XCI file to use for rendering. If the root is not specified, the file is assumed to reside in the location where the AEM Forms EAR files are deployed. For example, in a JBoss turnkey installation, you can see default XCI files in the *[Install folder]*/jboss/server/lc_turnkey/svcdata/XMLFormService folder. *[Install folder]* is the location where the AEM Forms is installed. For information about creating a custom XCI file, see Designer Help

.

If you provide a literal value, clicking the ellipsis button opens the Select Asset dialog box. (See About-Select Asset.)

*Form Model Processing*

A FormModel value that specifies where scripts that are embedded into the form are executed.

If you use a literal value, select one of these values:

Use Form Template Default:

(Default) The Forms service checks the form design to determine whether to render the form on the client or the server.

Client Side:

The form is rendered on the client. Do not use scripts that run on the server. When this value is selected, scripts that run on the server generate a warning on the AEM Forms Server.

Server Side:

The form is rendered on the server.

Both Server and Client side:

The form is rendered on both the server and the client.

*Attachments*

A map value of document values that specifies the attachments that are rendered with the PDF form. You cannot use a literal value.

**Output properties**

Properties to store the rendered PDF form.

*Rendered Form*

The location in the process data model to store the rendered PDF form. The data type is document.

**Additional Output properties**

Additional properties to store the rendered PDF form and the results of rendering the form.

*Page Count*

The location in the process data model to store the number of pages in the PDF form. The data type is long.

*Locale*

The location in the process data model to store the locale code of the rendered PDF form. The data type is string.

*Forms Result*

The location in the process data model to store the all the output results as a combined complex value. The data type is FormsResult.

**Exceptions**

This operation throws a RenderFormException exception when an error occurs rendering a PDF form.

## renderPDFForm operation(deprecated)

**NOTE:** This operation is deprecated. Use the renderPDFForm operation instead. It is recommended that when you upgrade a process, you change it to use the renderPDFForm operation. Only use the renderPDFform(deprecated) operation when you cannot reference a form design or form data using a `document` object. (See About deprecatedoperations.)

Retrieves the specified form, merges form data, and transforms it to a PDF form for a client application, such as a Adobe Reader. Use this operation when you want to use form designs or form data from a network, local file system, or HTTP location as literal values.

For example, your application must render a PDF form to a user for them to digitally sign in Acrobat. A PDF form has embedded security features and the ability for users to digitally sign the form. When the form design to render is on the network, use the renderPDFForm (deprecated) operation to render an interactive PDF form.

For information about the General and Route Evaluation property groups, see Commonoperation properties.

**Input properties**

The properties that specify the form design, form data, and options when rendering a PDF form.

*Form To Render*

A string value that specifies the name of the form design. This value is combined with the value of the Content Root URI property in this operation to construct an absolute path to the form.

If you provide a literal value, the name of the form must be typed in the box below the Form To Render property.

For example, a folder in the repository named *form designs* contains the form design named *form.xdp*. To access the form design, type `/form.xdp` in the Form To Render property and `repository:///formdesigns` in the Content Root URI property. The absolute path created is repository:///formdesigns/form.xdp, which is required to access the form design.

*Form Data*

A document value that represents the data to merge with the form during rendering. Form data that is provided as XML must be deserialized by using the `deserialize` function. (See deserialize.)

If you provide a literal value, clicking the ellipsis button opens the Select Asset dialog box. (See About-Select Asset.)

*PDF Form Render Options*

A PDFFormRenderSpec value that represents run-time options for rendering a PDF form.

If you provide a literal value, set the following options:

Character Set:

Sets the character set used to encode in the rendered PDF form. Select the character set to use or select one of these values:

Use Server Default:

(Default) Use the Character Set setting that is configured on the AEM Forms Server. The Character Set setting is configured using administration console. (See Forms administrationhelp

.)

Use Custom Value:

Use a character set that is not available in the list. After selecting this value, in the text box beside the list, and type the canonical name (Java.nio API) of the encoding set to use. For a list of character sets, see http://java.sun.com/j2se/1.5.0/docs/guide/intl/encoding.doc.html

.

Locale:

Sets the language used to send validation messages to client devices, such as web browsers, when an HTML form is rendered. Select select a language from the list or select one of these values:

Use Server Default:

*(Default) Use the Locale setting that is configured on the AEM Forms Server. The Locale setting is configured using administration console. (See Forms administrationhelp

.)

**Use Custom Value:**

Use a locale that is not available in the list. After selecting this value, in the text box beside the list, type the Locale ID of the locale code to use. For a list of supported locale codes, see http://java.sun.com/j2se/1.5.0/docs/guide/intl/locale.doc.html

.

Cache Form On Server:

Sets whether the form design is cached on the server. Caching forms on the server improves performance for forms that are rendered on the server. No default value is provided. When no value is selected, the Form Cache Control settings are used. The Form Cache Control Settings are configured using Administration Console. (See Forms administrationhelp

.) Select one of these values:

False:

The form design is not cached on the server.

True:

The form design is cached on the server.

Acrobat Version:

Sets the minimum Acrobat and Adobe Reader version required to open and modify the rendered PDF form. Select one of these values:

Use Form Template Default:

(Default) The Target Version setting in the form design determines the minimum version of Acrobat or Adobe Reader. In addition, the form design determines the PDF Version.

Acrobat and Adobe Reader 6 or later:

Acrobat or Adobe Reader 6 and later can open the PDF form. PDF Version 1.5 is used.

Acrobat and Adobe Reader 7.0 or later:

Acrobat or Adobe Reader 7 and later can open the PDF form. PDF Version 1.6 is used.

Acrobat and Adobe Reader 7.0.5 or later:

Acrobat or Adobe Reader 7.0.5 and later can open the PDF form. PDF Version 1.65 is used.

Acrobat and Adobe Reader 8 or later:

Acrobat or Adobe Reader 8 and later can open the PDF form. PDF Version 1.7 is used.

Acrobat and Adobe Reader 8.1 or later:

or Adobe Reader 8.1 and later can open the PDF form. PDF Version 1.7-ADBE-1 is used.

Acrobat and Adobe Reader 9 or later:

Acrobat or Adobe Reader 9 and later can open the PDF form. PDF Version 1.7-ADBE-3 is used.

Populate XML Data:

Sets whether the XML data is produced from the form design based on its current processing state. Select one of these values:

False:

(Default) Do not produce the XML data.

True:

Produce the XML data.

Tagged PDF:

Sets whether to create a tagged Adobe PDF form. A tagged PDF form defines a set of standard structure types and attributes that support the extraction of page content and reuse for other purposes. It is intended for use by client applications that perform the following types of operations:

- Simple extraction of text and graphics for pasting into other applications.
- Automatic reflow of text and associated graphics to fit a page of a different size than was assumed for the original layout.
- Processing text for such purposes as searching, indexing, and spell-checking.
- Conversion to other common file formats (such as HTML, XML, and RTF) with document structure and basic styling information preserved.
- Making content accessible by screen reader software.

*Select one of these values:*

*False:*

*Do not render a tagged PDF form.*

*True:*

*(Default) Render a tagged PDF form.*

*Linearized PDF:*

*Sets whether to render a linearized PDF form. A linearized PDF form is organized so that it supports incremental access in a network environment. For example, a linearized PDF can be displayed in a web browser before the entire PDF document is downloaded. Select one of these values:*

*False:*

*(Default) Do not render a linearized PDF form. It is best to use this option for non-web applications.*

*True:*

*Render a linearized PDF form. It is best to use this option for optimized web applications.*

*Seed PDF:*

*Sets an initial PDF form that is used in a PDF transformation to optimize delivery. The seed PDF form specifies a customized PDF form containing only fonts that is appended with a form design and data. This property is used when the form is opened using Acrobat 7.0 or later. No default value is provided.*

*Render At Client:*

*Sets whether to enable the delivery of PDF content by using the client-side rendering capability of Acrobat 7.0 or Adobe Reader 7.0 and later. Client-side rendering improves the performance of the Forms service. This property applies only to PDF, PDFForm, or PDFMerge transformations. Select one of these values:*

*Use Server Default:*

*(Default) Use the Render At setting version on the server to open PDF forms rendered by the Forms service. The Render At setting is configured in Administration Console. (See Forms administrationhelp*

*.)*

*Use Form Template Default:*

*The Forms service determines the form rendition based on the setting in the form design.*

*Yes:*

*A dynamic PDF form is generated and rendering occurs in Acrobat. Rendering of a dynamic form occurs only on Acrobat 7.0 or later. No rendering occurs for earlier version of Acrobat.*

*No:*

*A static PDF form is generated. Rendering does not occur on the client.*

*Stand Alone Rendition:*

*Sets whether the form can be rendered without state information. State information is used for rendering interactive forms that require user interaction with the server for submissions. Select one of these values:*

*False:*

*(Default) The form is rendered without state information and without embedded JavaScript that runs on the client web browser. The Forms service renders the form after server-side calculations are performed and the results are displayed in the rendered form. The results of the calculation are displayed in the rendered form. Because of the required interaction with the Forms service, the form cannot be used offline.*

*True:*

*The form is rendered with state information and embedded JavaScript. The JavaScript code runs on the client with no interaction with the server. In addition, the form can be used offline.*

*Form Model:*

*Sets the location where scripts that are embedded into the form are executed. Select one of these values:*

*Use Form Template Default:*

*(Default) The Forms service checks the form design to determine whether to render the form on the client or on the server.*

*Client Side:*

*The form is rendered on the client. Server-side scripts should not be used. Scripts in a form that are run on the server when this option is selected generate a warning on the AEM forms Server.*

*Server Side:*

*The form is rendered on the server.*

*Both Server and Client side:*

*The form is rendered on both the server and the client.*

*XCI URI:*

*Sets the URI location of the XCI file to use for rendering. If the root is not specified, the file is assumed to reside in the location where the AEM Forms EAR files are deployed.*

*Client Cache:*

*Sets whether the rendered PDF form is cached on the client web browser. Only forms that are rendered as interactive PDF forms can be stored in the client web browser cache.*

*When client caching is used, the timestamps of the cached PDF form is compared with the timestamp of the PDF form generated on the server. If the timestamps are the same, the PDF form is retrieved from the client cache. When the compared timestamps are different, the server redelivers the PDF form. Using the cache on the client results in reduced bandwidth usage and improves per-*

*formance. Performance improves because the Forms service does not have to redeliver the PDF form to the client application. Select one of these values:*

*False:*

*(Default) Do not cache the form on the client.*

*True:*

*Cache the form on the client.*

*Generate Server Appearance:*

*Sets whether the appearance for the PDF form is generated on the server. Appearances include the layout of fields and graphical elements in the PDF form. When you generate the appearance on the server, the form is rendered on the server and merged with data. Generate the appearance when you want to use the rendered PDF form in a subsequent operation. For example, if you need to apply a digital signature to the rendered PDF form on the server, generate the appearance on the server before applying the digital signature. If the appearance is not generated on the server, it is generated in Acrobat or Adobe Reader when the PDF form is rendered on the client, which invalidates the digital signature that was applied.*

*False:*

*(Default) Do not generate the appearance on the server.*

*True:*

*Generate the appearance of the rendered PDF form on the server.*

*URL Options*

A URLSpec value that specifies the URI run-time information required to render a PDF form.

If you provide a literal value, you can specify the following options:

Application Web Root:

Sets the URL representing the root location that is used to access application-specific web content. This value is combined with the value of the Target URL option to construct an absolute submit URL. No default value is provided.

Target URL:

Sets the URL to access a web service or Java servlet that receives data from the client application when a user submits the form. Setting a value in this option sets the target URL in the form design to the value specified by this property. If this option is not an absolute URL, it is combined with the value from the Application Web Root option to construct an absolute URL. No default value is provided.

For Workspace, the value is in the format `http://`*[server name]*`:`*[port]*`/workspace-server/submit`, where *[server name]* is the name of the server where AEM Forms is deployed and *[port]* is the port that the application server uses to provide HTTP access for client software. These are the default ports for the supported application servers.

JBoss:

`8080`

WebLogic:

`7001`

WebSphere:

`9080`

Content Root URI:

Sets the URI or an absolute reference, which specifies the location in the repository to retrieve a form design. This value is combined with the value of the Form To Render property in this operation to construct an absolute path to the form. No default value is provided.

You can use the following sources for a URI or absolute reference:

- Repository: The repository contains assets that you upload to the AEM Forms Server. The value `repository:///` references the root of the repository. The first two forward slashes are part of the protocol (repository://) and the third forward slash represents the root of the repository. For example, the documents folder is created below the root of the repository.
- Directory in the file system of the AEM Forms Server: You can specify a location on the AEM Forms Server, such as C:*[foldername]*. Using a location on the server is not recommended if you want to ensure portability of your application.
- Network directory: You can specify a location on the network, such as *[folder name].*
- Web location that is accessible by using HTTP: After you upload a file to a location on a web server, you can specify the location by using a URL, such as http://*[server name]*:*[port number]*/*[folder-name]*.

  *For example, a folder in the repository named form designs contains the form design named form.xdp. To access the form design, type `/form.xdp` in the Form To Render property and `repository:///formdesigns` in the Content Root URI property. The absolute path created is repository:///formdesigns/form.xdp, which is required to access the form design.*

  *Base URL:*

  *Sets the URL, which is the HTTP-equivalent of the Content Root URI. This value is required only when you render HTML forms (renderHTMLForm(deprecated) and renderHTMLForm operations) that include HREF references to external dependencies, such as images or scripts. When a dependency path is absolute, this value is ignored. No default value is provided.*

*Attachments*

A map value of document values that specifies the attachments that are rendered with the PDF form.

**Output properties**

Properties to store the rendered PDF form and the results of rendering the form.

*Rendered Form*

The location in the process data model to store the rendered PDF form. The data type is document.

*Page Count*

The location in the process data model to store the number of pages in the PDF form. The data type is long.

*Locale*

The location in the process data model to store the locale code of the rendered PDF form. The data type is string.

*Forms Result*

The location in the process data model to store the result of rendering the PDF form. The data type is FormsResult.

**Exceptions**

This operation throws a RenderFormException exception when an error occurs rendering a PDF form.

## Forms exceptions

The Forms service provides the following exception for throwing exception events.

**ProcessFormSubmissionException**

Thrown when an error occurs when processing a processing a form submission.

**RenderFormException**

Thrown when an error occurs rendering a form to a Guide, HTML, or PDF.

# 22.29. FTP

Enables processes to interact with an FTP server. FTP service operations can retrieve files from the FTP server, put files on the FTP server, and delete files from the FTP server. You can retrieve files and store them as process data, or as files on the file system of the AEM Forms Server or other file system that the server can access.

**NOTE:** The FTP service does not support FTP over SSH or FTP over SSL.

If you specify local paths to files or directories for any operation properties, the paths are interpreted as being on the file system of the AEM Forms Server.

**NOTE:** The user account that is used to run the AEM forms Server must have the required permissions to interact with the files and file locations that the service's operations target.

You can modify the default settings for the FTP service by using one of the following methods:

- In Workspace, modify settings for the FTP service in the Components view. (See Editing service configurations.)

- In the administration console, select Services > Applications and Services > Service Management and click the FTP service. (See Applications and Services AdministrationHelp
- .)

## FTP service configuration

The FTP service must be configured before the service operations can be used successfully. (See Editing service configurations.)

The following properties, used for connecting to the FTP server, can be configured.

**Default Host:**

The IP address or URL of the FTP server.

**Default Port:**

The port used to connect to the FTP server. The default is `21`.

**Default Username:**

The name of the user account that you can use to access the FTP server. The user account must have sufficient privileges to perform the FTP operations that this service provides.

**Default Password:**

The password to use with the user name specified for Default Username for authenticating with the FTP server.

**NOTE:** The FTP service does not support FTP over SSH or FTP over SSL.

## Delete file operation

Deletes a file from the FTP server.

For information about the General and Route Evaluation property groups, see Commonoperation properties.

### Remote File Details properties

Properties for defining the file to delete.

*Remote File Name*

A `string` value that represents the name of the file to delete. The file name does not include the path to the file. Specify the path using the Remote Directory Path property.

*Remote Directory Path*

A `string` value that represents the path to the file to delete. The path does not include the file name. Specify the file name using the Remote File Name property.

### Connection Settings properties

Use the Connection Settings properties to specify information required for connecting to the FTP server.

*Use Configuration Options*

A `boolean` value that specifies whether to use the settings in the service configuration instead of the settings specified in the properties of the operation. A value of `True` means to use the default connection settings configured for the FTP service. A value of `False` means to use the settings configured in the properties for the operation.

If you use a literal value to configure this property, selecting the Use Configuration Options check box means to use the default connection properties that are set for the FTP service. If you want to use the connection settings specified in the operation, deselect the Use Configuration Options check box

*Host*

A `string` value that represents the IP address or URI. For example, you can use the name of the computer that is resolved by Domain Naming Scheme (DNS).

*Port Number*

An `int` value that represents the port number for transferring files. The default port number is 21.

*User*

A `string` value that represents the user name to connect to the FTP server. No default is provided.

*Password*

A `string` value that represents the password to use for authenticating with the FTP server. No default is provided.

### Delete Operation Result properties

Properties that specify where to save the operation results.

*Result*

The location to save the operation result. The data type is `boolean`. A value of `True` indicates that the file was deleted. A value of `False` indicates that the file could not be deleted.

### Exceptions

The exception event attached to this operation can receive FTPConnectionException, FTPFileUnavailableException, IllegalArgumentException, and IOException exceptions.

## Get operation

Retrieves the contents of a file from the FTP server. You can save the contents as process data.

For information about the General and Route Evaluation property groups, see Commonoperation properties.

### Remote File Details properties

Properties for defining the file to retrieve.

*File Name On FTP Server*

A `string` value that represents the name of the file to retrieve. The file name does not include the path to the file. Specify the path using the Path On FTP Server property.

*Path On FTP Server*

A `string` value that represents the path to the file to retrieve. The path does not include the file name. Specify the file name using the File Name On FTP Server property.

### File Transfer Mode properties

Properties for specifying the format of transferred data.

*File Transfer Mode*

A `TransferModeEnum` value that represents the format of the file data that is transferred. Valid values are `Binary` and `ASCII`.

You should use binary transfer mode only if you are certain that the computer that runs the AEM Forms Server and the FTP server interpret ASCII characters in the same way. For example, different types of computer systems interpret the end-of-file character differently. Transferring text files from one computer type to another can result in nonsensical characters being inserted at the end of the file.

However, transferring files in ASCII mode is marginally faster than transferring in binary mode.

### Connection Settings properties

Use the Connection Settings properties to specify information required for connecting to the FTP server.

*Use Configuration Options*

A `boolean` value that specifies whether to use the settings in the service configuration instead of the settings specified in the properties of the operation. A value of `True` means to use the default connection settings configured for the FTP service. A value of `False` means to use the settings configured in the properties for the operation.

If you use a literal value to configure this property, selecting the Use Configuration Options check box means to use the default connection properties that are set for the FTP service. If you want to use the connection settings specified in the operation, deselect the Use Configuration Options check box

*Host*

A `string` value that represents the IP address or URI. For example, you can use the name of the computer that is resolved by Domain Naming Scheme (DNS).

*Port Number*

An `int` that represents the port number for transferring files. The default port number is 21.

*User*

A `string` value that represents the user name to connect to the FTP server. No default is provided.

*Password*

A `string` value that represents the password to use for authenticating with the FTP server. No default is provided.

**File Content properties**

Properties for handling transferred data.

*Content*

The location to save the transferred file data. The data type is `document`.

**Exceptions**

The exception event attached to this operation can receive FTPConnectionException, FTPFileUnavailableException, IllegalArgumentException, and IOException exceptions.

## Get list of files operation

Retrieves a list of files that reside in a directory on an FTP server.

For information about the General and Route Evaluation property groups, see Commonoperation properties.

**Remote Directory Details properties**

Properties that define the directory on the FTP server.

*Remote Directory Path*

A `string` value that represents the directory on the FTP server that contains the files that you want listed.

**Connection Settings properties**

Use the Connection Settings properties to specify information required for connecting to the FTP server.

*Use Configuration Options*

A `boolean` value that specifies whether to use the settings in the service configuration instead of the settings specified in the properties of the operation. A value of `True` means to use the default connec-

tion settings configured for the FTP service. A value of `False` means to use the settings configured in the properties for the operation.

If you use a literal value to configure this property, selecting the Use Configuration Options check box means to use the default connection properties that are set for the FTP service. If you want to use the connection settings specified in the operation, deselect the Use Configuration Options check box

*Host*

A `string` value that represents the IP address or URI. For example, you can use the name of the computer that is resolved by Domain Naming Scheme (DNS).

*Port Number*

An `int` that represents the port number for transferring files. The default port number is 21.

*User*

A `string` value that represents the user name to connect to the FTP server. No default is provided.

*Password*

A `string` value that represents the password to use for authenticating with the FTP server. No default is provided.

**Retrieved Files Detail properties**

Properties that define how to handle retrieved data.

*List Of File Info Objects*

The location to save the list of files that is retrieved. The data is a `list` of sub-type `FTP FileInfo` data items. For information about accessing data in collections, see Accessingdata in data collections.

**Exceptions**

The exception event attached to this operation can receive FTPConnectionException, FTPFileUnavailableException, IllegalArgumentException, and IOException exceptions.

## Get multiple documents operations

Retrieves multiple files from the FTP server based on a file name pattern.

For information about the General and Route Evaluation property groups, see Commonoperation properties.

**Remote Files Options properties**

Properties for defining the files to retrieve.

*File Pattern For Remote Files*

A `string` value that represents the a textual pattern that matches the name of the files that you want to retrieve from the server. The file pattern can use the asterisk (*) and question mark (?) symbols:

- Use the asterisk (*) to represent any series of characters. For example, the pattern te*.xdp matches the file name test.xdp.
- Use the question mark (?) to represent any single character. For example, the pattern te?t.xdp matches the file name test.xdp.

*Remote Directory Path*

A `string` value that represents the path to the directory that contains the files that you want to retrieve.

*File Transfer Mode*

A `TransferModeEnum` value that represents the format of the file data that is transferred. Valid values are `Binary` and `ASCII`.

You should use binary transfer mode only if you are certain that the computer that runs the AEM Forms Server and the FTP server interpret ASCII characters in the same way. For example, different types of computer systems interpret the end-of-file character differently. Transferring text files from one computer type to another can result in nonsensical characters being inserted at the end of the file.

However, transferring files in ASCII mode is marginally faster than transferring in binary mode.

**Connection Settings properties**

Use the Connection Settings properties to specify information required for connecting to the FTP server.

*Use Configuration Options*

A `boolean` value that specifies whether to use the settings in the service configuration instead of the settings specified in the properties of the operation. A value of `True` means to use the default connection settings configured for the FTP service. A value of `False` means to use the settings configured in the properties for the operation.

If you use a literal value to configure this property, selecting the Use Configuration Options check box means to use the default connection properties that are set for the FTP service. If you want to use the connection settings specified in the operation, deselect the Use Configuration Options check box

*Host*

A `string` value that represents the IP address or URI. For example, you can use the name of the computer that is resolved by Domain Naming Scheme (DNS).

*Port Number*

An `int` that represents the port number for transferring files. The default port number is 21.

*User*

A `string` value that represents the user name to connect to the FTP server. No default is provided.

*Password*

A `string` value that represents the password to use for authenticating with the FTP server. No default is provided.

**Retrieved Documents properties**

Properties that specify how to handle retrieved data.

*List Of Documents*

The location to save the retrieved files. The data is a `list` of `document` values.

**Exceptions**

The exception event attached to this operation can receive FTPConnectionException, FTPFileUnavailableException, IllegalArgumentException, and IOException exceptions.

## Get to file system operation

Retrieves a file from the FTP server and saves it to the file system of the AEM Forms Server. Can also save the file to a shared folder on another machine that the AEM Forms Server can access.

For information about the General and Route Evaluation property groups, see Commonoperation properties.

**Local File Details properties**

Properties that define the location to save the retrieved file.

*Local Directory Path*

A `string` value that represents the path to the directory on the AEM Forms Server where you want to save retrieved files.

**Remote File Details properties**

Properties for defining the file to retrieve.

*File Name On FTP Server*

A `string` value that represents the name of the file to retrieve. The file name does not include the path. Specify the path using the Path On FTP Server property.

*Path On FTP Server*

A `string` value that represents the path to the file to retrieve. The path does not include the file name. Specify the file name using the File Name On FTP Server property.

**File Transfer Mode properties**

Properties that specify details about data transfer.

*File Transfer Mode*

A `TransferModeEnum` value that represents the format of the file data that is transferred. Valid values are `Binary` and `ASCII`.

You should use binary transfer mode only if you are certain that the computer that runs the AEM Forms Server and the FTP server interpret ASCII characters in the same way. For example, different types of computer systems interpret the end-of-file character differently. Transferring text files from one computer type to another can result in nonsensical characters being inserted at the end of the file.

However, transferring files in ASCII mode is marginally faster than transferring in binary mode.

**Connection Settings properties**

Use the Connection Settings properties to specify information required for connecting to the FTP server.

*Use Configuration Options*

A `boolean` value that specifies whether to use the settings in the service configuration instead of the settings specified in the properties of the operation. A value of `True` means to use the default connection settings configured for the FTP service. A value of `False` means to use the settings configured in the properties for the operation.

If you use a literal value to configure this property, selecting the Use Configuration Options check box means to use the default connection properties that are set for the FTP service. If you want to use the connection settings specified in the operation, deselect the Use Configuration Options check box

*Host*

A `string` value that represents the IP address or URI. For example, you can use the name of the computer that is resolved by Domain Naming Scheme (DNS).

*Port Number*

An `int` that represents the port number for transferring files. The default port number is 21.

*User*

A `string` value that represents the user name to connect to the FTP server. No default is provided.

*Password*

A `string` value that represents the password to use for authenticating with the FTP server. No default is provided.

**File Transfer Result properties**

Properties that specify where to save the operation results.

*Result*

The location to save the operation result. The data type is `boolean`. A value of `True` indicates that the file was transferred. A value of `False` indicates that the file could not be transferred.

**Exceptions**

The exception event attached to this operation can receive FTPConnectionException, FTPFileUnavailableException, IllegalArgumentException, and IOException exceptions.

## Put operation

Uploads process data to a directory on the FTP server and saves the data as a file.

For information about the General and Route Evaluation property groups, see Commonoperation properties.

**File Details properties**

Properties that define the data to upload.

*File Content*

A The content should be type of `document`, `xml`, `string` or `byte` value that represents a location in the process data model that contains the data that you want to upload to the FTP server.

**Remote File Details properties**

*Path On FTP Server*

A `string` value that represents the path on the FTP server where you want to save the file. The path does not include the file name. Specify the file name using the File Name On FTP Server property. The path must contain a closing slash, for example, /temp/input/.

*File Name On FTP Server*

A `string` value that represents the name of the file to upload. The file name does not include the path. Specify the path using the Path On FTP Server property. You should include the file name extension if the file consumers need to associate the file with software to open it.

### File Transfer Mode properties

Properties that specify details about data transfer.

*File Transfer Mode*

A `TransferModeEnum` value that represents the format of the file data that is transferred. Valid values are `Binary` and `ASCII`.

You should use binary transfer mode only if you are certain that the computer that runs the AEM Forms Server and the FTP server interpret ASCII characters in the same way. For example, different types of computer systems interpret the end-of-file character differently. Transferring text files from one computer type to another can result in nonsensical characters being inserted at the end of the file.

However, transferring files in ASCII mode is marginally faster than transferring in binary mode.

### Connection Settings properties

Use the Connection Settings properties to specify information required for connecting to the FTP server.

*Use Configuration Options*

A `boolean` value that specifies whether to use the settings in the service configuration instead of the settings specified in the properties of the operation. A value of `True` means to use the default connection settings configured for the FTP service. A value of `False` means to use the settings configured in the properties for the operation.

If you use a literal value to configure this property, selecting the Use Configuration Options check box means to use the default connection properties that are set for the FTP service. If you want to use the connection settings specified in the operation, deselect the Use Configuration Options check box

*Host*

A `string` value that represents the IP address or URI. For example, you can use the name of the computer that is resolved by Domain Naming Scheme (DNS).

*Port Number*

An `int` that represents the port number for transferring files. The default port number is 21.

*User*

A `string` value that represents the user name to connect to the FTP server. No default is provided.

*Password*

A `string` value that represents the password to use for authenticating with the FTP server. No default is provided.

### File Transfer Result properties

Properties that specify where to save the operation results.

*Result*

The location to save the operation result. The data type is `boolean`. A value of `True` indicates that the file was transferred. A value of `False` indicates that the file could not be transferred.

**Exceptions**

The exception event attached to this operation can receive FTPConnectionException, FTPFileUnavailableException, IllegalArgumentException, and IOException exceptions.

## Put from file system operation

Uploads a file from the file system of the AEM Forms Server to a directory on the FTP server. Can also upload the file from a shared folder on another machine that the AEM Forms Server can access.

For information about the General and Route Evaluation property groups, see Commonoperation properties.

**File Details properties**

Properties that identify the file to upload.

*File Name On Local File System*

A `string` value that represents the name of the file to upload to the FTP server.

*Directory On Local File System*

A `string` value that represents the directory on the file system where the file to upload is located.

**Remote Path properties**

Properties that identify where to upload the file

*Path On FTP Server*

A `string` value that identifies the path to the directory on the FTP server where you want to save the uploaded file.

**File Transfer Mode properties**

Properties that specify details about data transfer.

*File Transfer Mode*

A `TransferModeEnum` value that represents the format of the file data that is transferred. Valid values are `Binary` and `ASCII`.

You should use binary transfer mode only if you are certain that the computer that runs the AEM Forms Server and the FTP server interpret ASCII characters in the same way. For example, different types of

computer systems interpret the end-of-file character differently. Transferring text files from one computer type to another can result in nonsensical characters being inserted at the end of the file.

However, transferring files in ASCII mode is marginally faster than transferring in binary mode.

### Connection Settings properties

Use the Connection Settings properties to specify information required for connecting to the FTP server.

*Use Configuration Options*

A `boolean` value that specifies whether to use the settings in the service configuration instead of the settings specified in the properties of the operation. A value of `True` means to use the default connection settings configured for the FTP service. A value of `False` means to use the settings configured in the properties for the operation.

If you use a literal value to configure this property, selecting the Use Configuration Options check box means to use the default connection properties that are set for the FTP service. If you want to use the connection settings specified in the operation, deselect the Use Configuration Options check box

*Host*

A `string` value that represents the IP address or URI. For example, you can use the name of the computer that is resolved by Domain Naming Scheme (DNS).

*Port Number*

An `int` that represents the port number for transferring files. The default port number is 21.

*User*

A `string` value that represents the user name to connect to the FTP server. No default is provided.

*Password*

A `string` value that represents the password to use for authenticating with the FTP server. No default is provided.

### File Transfer Result properties

Properties that specify where to save the operation results.

*Result*

The location to save the operation result. The data type is `boolean`. A value of `True` indicates that the file was transferred. A value of `False` indicates that the file could not be transferred.

### Exceptions

The exception event attached to this operation can receive FTPConnectionException, FTPFileNotFoundException, FTPFileUnavailableException, IllegalArgumentException, and IOException exceptions.

## Put multiple documents operation

Uploads one or more `document` values to the FTP server. The document values must be stored in a `list` or `map` value.

For information about the General and Route Evaluation property groups, see Commonoperation properties.

### File Options properties

Properties that identify the documents to upload to the FTP server.

*Documents*

A `list` or `map` value that holds one or more document values that you want to upload to the FTP server.

If you specify a `map` value, the keys are used as the file names for the documents.

If you use a `list` value, the file names are based on the properties of the `document` values in the `list`.

*Remote Directory Path*

A `string` value that represents the directory on the FTP server where you want to save the uploaded files. The path must contain a closing slash, for example, /temp/input/.

*File Transfer Mode*

A `TransferModeEnum` value that represents the format of the file data that is transferred. Valid values are `Binary` and `ASCII`.

You should use binary transfer mode only if you are certain that the computer that runs the AEM Forms Server and the FTP server interpret ASCII characters in the same way. For example, different types of computer systems interpret the end-of-file character differently. Transferring text files from one computer type to another can result in nonsensical characters being inserted at the end of the file.

However, transferring files in ASCII mode is marginally faster than transferring in binary mode.

### Connection Settings properties

Use the Connection Settings properties to specify information required for connecting to the FTP server.

*Use Configuration Options*

A `boolean` value that specifies whether to use the settings in the service configuration instead of the settings specified in the properties of the operation. A value of `True` means to use the default connection settings configured for the FTP service. A value of `False` means to use the settings configured in the properties for the operation.

If you use a literal value to configure this property, selecting the Use Configuration Options check box means to use the default connection properties that are set for the FTP service. If you want to use the connection settings specified in the operation, deselect the Use Configuration Options check box

*Host*

A `string` value that represents the IP address or URI. For example, you can use the name of the computer that is resolved by Domain Naming Scheme (DNS).

*Port Number*

An `int` that represents the port number for transferring files. The default port number is 21.

*User*

A `string` value that represents the user name to connect to the FTP server. No default is provided.

*Password*

A `string` value that represents the password to use for authenticating with the FTP server. No default is provided.

**Result properties**

Properties that specify where to save the operation results.

*Result*

The location to save the operation result. The data type is `boolean`. A value of `True` indicates that the file was transferred. A value of `False` indicates that the file could not be transferred.

**Exceptions**

The exception event attached to this operation can receive FTPConnectionException, FTPFileUnavailableException, IllegalArgumentException, and IOExceptionexceptions.

## -FTP Exceptions

The FTP service provides the following exceptions for throwing exception events.

**FTPConnectionException**

Thrown when a file that is targeted by an operation cannot be found on the file system of the AEM Forms Server.

**FTPFileNotFoundException**

Thrown when a file that is targeted by an operation cannot be found on the file system of the AEM Forms Server.

### FTPFileUnavailableException

Thrown when a file that is targeted by an operation is not available on the FTP server. The user account that you are using to connect to the FTP server may not have permissions to perform the operation, or the file cannot be located on the FTP server.

### IllegalArgumentException

Thrown when an input property for an operation is an unexpected value or not specified. For example, for Put Multiple Documents, the Documents property should be type `map` or `list`. An IllegalArgument-Exception is thrown if the value is type `string` or not specified.

### IOException

Thrown when a file input/output error occurs on the FTP server.

## 22.30. Generate PDF

The Generate PDF service converts native file formats to PDF. It also converts PDF to other file formats and optimizes the size of PDF documents.

The Generate PDF service uses native applications to convert the following file formats to PDF. Unless otherwise indicated, only the German, French, English, and Japanese versions of these applications are supported. *Windows only* indicates support for only Windows Server 2008.

- Microsoft Office 2007 and 2010 to convert DOC, DOCX, RTF, TXT, XLS, XLSX, PPT, PPTX, VSD, MPP, XPS (Windows only), and PUB (Windows only)
- Autodesk AutoCAD DWF and DWG (English only)
- OpenOffice 3.3 to convert ODT, ODS, ODP, ODG, ODF, SXW, SXI, SXC, SXD, DOC, DOCX, RTF, TXT, XLS, XLSX, PPT, and PPTX. (Windows, Linux, and Solaris only)

  **NOTE:** The Generate PDF service does not support the 64-bit versions of OpenOffice

  **NOTE:** PDF Generator does not support DWF and DWG files created with AutoCAD 2013.
- Adobe FrameMaker® 8 to convert FM (Windows only)
- Adobe PageMaker® 7.0 to convert PMD, PM6, P65, and PM (Windows only)
- Native formats supported by third-party applications (requires development of setup files specific for the application) (Windows only)

The Generate PDF service converts the following standards-based file formats to PDF.

- Video formats: SWF, FLV (Windows only)
- Image formats: JPEG, JPG, JP2, J2Kí, JPC, J2C, GIF, BMP, TIFF, TIF, PNG, JPF
- HTML

**NOTE:** AEM Forms supports only 32-bit editions of the above mentioned software.

The Generate PDF service converts PDF to the following file formats (Windows only):

- Encapsulated PostScript (EPS)

- HTML 3.2
- HTML 4.01 with CSS 1.0
- DOC (Microsoft Word format)
- RTF
- Text (both accessible and plain)
- XML
- PDF/A-1b and PDF/A-1a that use only the DeviceRGB color space
- PDF/E-1 that uses only the DeviceRGB color space
- DOCX
- XLSX
- PPTX

**NOTE:** The PDF conversion of FM, PMD, PM6, P65, PM, DWG, MPP, SWF, XPS, FLV, VSD, WordPerfect documents, and Microsoft Office documents (DOC, XLS, PPT, WPD, MPP, RTF, TXT) is possible only if Acrobat XI Pro is installed.

The Generate PDF service requires that you perform these administrative tasks:

- Install required native applications on the computer hosting AEM Forms
- Install Adobe Acrobat DC Pro on the computer hosting AEM forms
- Perform post-installation setup tasks

These tasks are described in Installing and Deploying AEM Forms UsingJBoss Turnkey

.

For information about using the Generate PDF service, see Services Reference for AEM Forms

.

## Generate PDF service configuration

The following properties can be configured for the Generate PDF service.

**Adobe PDF Settings:**

The name of the pre-configured Adobe PDF settings to apply to a conversion. The Adobe PDF settings are configured in administration console. (See Configuring Adobe PDF settings in AEM Forms administration help

.)

**Security Settings:**

The name of the pre-configured security settings to apply to a conversion job. The security settings are configured in Administration Console. (See Configure security settings in AEM Forms administration help

.)

**File type Settings:**

The name of the pre-configured File Type Setting to apply to a conversion job. The file type settings are configured in Administration Console. (See Configuring file type settings in AEM Forms administration help

.)

**Use Acrobat WebCapture (Windows Only):**

When this property is `true`, the Generate PDF service uses Acrobat Pro Extended 9.0 for all conversions from HTML to PDF. Using Acrobat Pro Extended can improve the quality of the PDF files, though performance can be slightly lower. The default value is `false`.

**Use Acrobat Image Conversion (Windows Only):**

When this setting is true, theGenerate PDF service uses Acrobat Pro Extended 9.0 for all conversions from image to PDF. This setting is useful only if the Generate PDF service's conversion mechanism cannot successfully convert a significant proportion of input images. The default value is `false`.

**Enable Acrobat-based AutoCAD Conversions (Windows Only):**

When this setting is true, the Generate PDF service uses Acrobat Pro Extended 9.0 for all conversions from DWG to PDF. This setting is useful only if AutoCAD is not installed on the server or the AutoCAD conversion mechanism is unable to convert files successfully.

**Regular Expressions For Finding Out Prohibited Special Characters In User Name (Windows Only):**

A regular expression that specifies characters that interfere with the Export PDF and Optimize PDF operations when the characters appear in a user's name. The following default value removes $, \, /, ?, @, *, +, ", |, :, ;, =, >, <, [, and ] characters:

```
[\\$,\\,\/,\?,\@,\*,\+,\",\|,\:,\;\,\=,\>,\<,\[,\]]
```

**ImageToPDF Pool Size:**

The pool size of the Generate PDF service's Image-to-PDF converter. This number represents the maximum simultaneous Image-to-PDF conversions that the Generate PDF service performs. The default value of 3 is recommended for single-processor systems. Increase this value for multi-processor systems.

**HTML to PDF Pool Size:**

The pool size of the Generate PDF service's HTML-to-PDF converter. This number represents the maximum simultaneous HTML-to-PDF conversions that the Generate PDF service performs. The default value of 3 is recommended for single-processor systems. Increase this value for multi-processor systems.

**Fallback font family for HTML to PDF conversions:**

The font to use in PDF documents when the font used in the original HTML is not available to the AEM Forms Server. Specify a font if you expect to convert HTML pages that use unavailable fonts. For example, pages authored in regional languages could use unavailable fonts.

## CreatePDF operation

Converts different file formats into PDF documents. The supported file formats are Microsoft Word, Microsoft PowerPoint, Microsoft Excel, Microsoft Project, Microsoft Visio, Microsoft Publisher, and AutoCAD. In addition, any third-party generic application type for which you have a PDF generating application can be converted.

The file name of the PDF that is created must be provided as input data. The file name can be specified in either of the following locations:

- The `file` attribute of the InputDocument property
- The FilenameWith Extension property

Security permissions, PDF output settings, and metadata information can optionally be applied to the resultant PDF document.

For information about the General and Route Evaluation property groups, see Commonoperation properties.

### Input properties

Properties to specify the file to convert and the conversion parameters.

*Input Document*

A `document` value that represents the file to convert.

If no value is provided for the Filename with Extension property, the `file` attribute of this `document` value must be set to the name of the input file. To set the `file` attribute, use the setDocAttribute XPath function.

**TIP:** Use the `setDocAttribute` function in an execute operation that the SetValue *service provides. The execute operation must be used before the CreatePDF operation on the process diagram.*

To provide a literal value, click the ellipsis button and select an asset from an application.

### Conversion Options properties

Properties for specifying how to convert the file to PDF.

*Filetype Settings*

(Optional) A `string` value that represents the settings to apply to the generated PDF document. The following values are provided by default:

- `Standard`
- `Standard OCR`
- `Standard OpenOffice`

You can create custom file type settings using the administration console. (See Generate PDF administration help

.) To specify a literal value, select a setting name from the list. The names of available settings that have been created using Administration Console appear in the list.

*Adobe PDF Settings*

(Optional) A `string` value that represents the PDF output settings to apply. The following values are valid by default:

- `High Quality Print`
- `Oversized Pages`
- `PDFA1b 2005 CMYK`
- `PDFA1b 2005 RGB`
- `PDFX1a 2001`
- `PDFX3 2003`
- `Press Quality`
- `Smallest File Size`
- `Standard`

**TIP:** You can create custom settings using the administration console and specify those settings here. (See Generate PDF administration help
*.)*

To specify a literal value, select a setting name from the list. The names of available settings that have been created using Administration Console appear in the list.

**Advanced Options properties**

Properties for specifying optional properties for the generated PDF document.

*File Extension*

(Optional) A `string` value that represents the file name, with the appropriate extension, of the document to convert. Provide this value only if the `file` attribute is not set for the `document` value specified for Input Document.

*Security Settings*

(Optional) A `string` value that represents the security settings. The only available default value is `No Security`. You can create custom settings using the administration console.

To specify a literal value, select a setting name from the list. The names of available settings that have been created using administration console appear in the list.

*Settings Document*

(Optional) A `document` value that contains settings to apply while generating the PDF document. For example, these settings can optimize the PDF for viewing on the web. Some settings are applied after the PDF document is created, such as initial view and security settings.

This XML file must conform to the XML Schema Definition (XSD) for Generate PDF configuration files.

To see an example XML configuration file, use Administration Console to export one. (See Importing and exporting PDF Generator configuration files in AEM Forms administration help

.)

To provide a literal value, click the ellipsis button and select an asset from an application.

*XMP Document*

(Optional) A `document` value that represents the file that contains metadata information to apply to the generated PDF document. Only UTF-8 encoded XMP metadata is supported.

To provide a literal value, click the ellipsis button and select an asset from an application.

**Output properties**

Property for saving the output PDF document.

*Created Document*

The location to store the output PDF document. The data type is `document`.

**Optional Output properties**

Property for saving a log of the service activity.

*Log Document*

The location to store an optional log file. The log file is a plain text file that contains error or diagnostic messages. The log file is not generated under all circumstances. The presence of a log file is not an indication of an error in the conversion because a log file could contain only diagnostic messages. The data type is `document`.

**Exceptions**

This operation can throw ConversionException, FileFormatNotSupportedException, and InvalidParameterException exceptions.

## ExportPDF operation

Converts a PDF document to a different file type according to the supported applications, which is based on the file name extensions. This operation is supported on Windows platforms only.

The file name of the PDF that is created must be provided as input data. The file name can be specified in either of the following locations:

- The `file` attribute of the InputDocument property
- The FilenameWith Extension property

For information about the General and Route Evaluation property groups, see Commonoperation properties.

**Input properties**

Properties to specify a PDF document and the required output format.

*Input Document*

A `document` value that represents the PDF document to convert. If no value is provided for the Filename with Extension property of this operation, the `file` attribute of this `document` value must be set to the name of the input file, such as `input.pdf`.

To set the `file` attribute, you use the setDocAttribute XPath function.

**TIP:** Use the `setDocAttribute` function in an execute operation that the SetValue *service provides. The execute operation must be used before the ExportPDF2 operation in the process diagram.*

To provide a literal value, click the ellipsis button and select an asset from an application.

*File Extension*

(Optional) A `string` value that represents the name of the input document. Type a value such as `.doc`, `.xls`, or `.txt`. Provide this value only if the `file` attribute is not set for the `document` value specified for Input Document.

*Format Type*

(Optional) A `string` value that represents the target file format to which the PDF document is converted. No default is provided. The following values can be specified:

- `EPS`
- `HTML32`
- `HTML40`
- `DOC`
- `RTF`
- `Accessible-Text`
- `Plain-Text`
- `XML10`
- `PDF/A-1a(sRGB)`
- `PDF/A-1b(sRGB)`
- `PDF/E-1(sRGB)`

To specify a literal value, select a format from the list.

*Settings Document*

(Optional) A `document` value that represents the configuration file that contains the settings to apply to the target document. It is typically an XML file.

This XML file must conform to the XML Schema Definition (XSD) for Generate PDF configuration files.

To see an example XML configuration file, use administration console to export one. (See Importing and exporting PDF Generator ES2 configuration files in AEM Forms administration help

.)

To provide a literal value, click the ellipsis button and select an asset from an application.

### Output properties

Properties to specify the converted document.

*Converted Document*

The location to store the exported document. The data type is `document`.

### Exceptions

This operation can throw ConversionException, FileFormatNotSupportedException, and InvalidParameterException exceptions.

## HtmlFileToPDF operation

Creates a PDF document from an HTML file or archived file (ZIP).

For information about the General and Route Evaluation property groups, see Commonoperation properties.

### Input properties

Properties to specify the HTML file and the conversion settings.

*Input Document*

A `document` value that represents the HTML or ZIP file to convert to a PDF document.

To provide a literal value, click the ellipsis button and select an asset from an application.

*Filetype Settings*

(Optional) A `string` value that represents the settings for handling how a conversion of supported file types occurs. The default is `Standard`. The following values are valid by default:

- `Standard`
- `Standard OCR`
- `Standard OpenOffice`

Custom file type settings can be created in administration console. (See Generate PDF administration help

.) To specify a literal value, select a setting name from the list. The names of available settings that have been created using Administration Console appear in the list.

*Security Settings*

(Optional) A `string` value that represents the security settings for the generated PDF document. The default is `No Security`. You can also create custom security settings using the administration console. (See Generate PDF administration help

.)

*Settings Document*

(Optional) A `document` value that represents the file that contains settings to apply while generating the PDF document. For example, these settings can optimize the PDF for viewing on the web. Some settings are applied after the PDF document is created, such as initial view and security settings.

This XML file must conform to the XML Schema Definition (XSD) for Generate PDF configuration files.

To see an example XML configuration file, use administration console to export one. (See Importing and exporting PDF Generator ES2 configuration files in AEM Forms administration help

.)

To provide a literal value, click the ellipsis button and select an asset from an application.

*XMP Document*

(Optional) A `document` value that contains metadata information to apply to the generated PDF document. Only UTF-8 encoded XMP metadata is supported.

To provide a literal value, click the ellipsis button and select an asset from an application.

### Output properties

Properties for specifying the converted document and the result of the operation.

*Created Document*

The location to store the resulting PDF document. The data type is `document`.

### Exceptions

This operation can throw ConversionException, FileFormatNotSupportedException, and InvalidParameterException exceptions.

## Html URL To PDF operation

Creates a PDF document from an HTML file that is located at a given URL.

For information about the General and Route Evaluation property groups, see Commonoperation properties.

### Input properties

Properties to specify the HTML file and the conversion settings.

*Input URL*

A `string` value that represents the URL of the HTML file to convert.

*Filetype Settings*

(Optional) A `string` value that represents the settings for handling how a conversion of supported file types occurs. The default is `Standard`. The following values are valid by default:

- `Standard`
- `Standard OCR`
- `Standard OpenOffice`

Custom file type settings can be created in administration console. (See Generate PDF administration help

.) To specify a literal value, select a setting name from the list. The names of available settings that have been created using Administration Console appear in the list.

*Security Settings*

(Optional) A `string` value that represents the security settings for the generated PDF document. The single available option is `No Security`. You can also create custom settings using the administration console.

*Settings Document*

(Optional) A `document` value that represents the file that contains settings to use for generating the PDF document. For example, you can apply settings for optimizing the PDF for viewing on the web. Some settings are applied after the PDF document is created, such as initial view and security settings.

This XML file must conform to the XML Schema Definition (XSD) for Generate PDF configuration files.

To see an example XML configuration file, use administration console to export one. (See Importing and exporting PDF Generator ES2 configuration files in AEM Forms administration help

.)

To provide a literal value, click the ellipsis button and select an asset from an application.

*XMP Document*

(Optional) A `document` value that contains metadata information to apply to the generated PDF document. Only UTF-8 encoded XMP metadata is supported.

To provide a literal value, click the ellipsis button and select an asset from an application.

**Output properties**

Property for specifying the converted document.

*Created Document*

The location to store the output PDF document. The data type is `document`.

### Exceptions

This operation can throw ConversionException, FileFormatNotSupportedException, and InvalidParameterException exceptions.

## Optimize PDF operation

Optimizes a PDF document by reducing the file size. This operation can also convert the PDF document to a different PDF version. This operation supports the same optimization settings that Adobe Acrobat supports.

For information about the General and Route Evaluation property groups, see Commonoperation properties.

### Input properties

Properties for specifying the PDF document to optimizes well as the optimization settings.

*Input document*

A `document` value that represents the PDF document to optimize.

To provide a literal value, click the ellipsis button and select an asset from an application.

*Filetype settings*

A `string` value that represents the name of the Generate PDF File Type settings to use. The AEM Forms Server administrator uses administration console to configure the conversion settings. (See Configuring file type settings in AEM Forms administration help

.) The following setting names are provided by default:

- Standard
- Standard OCR
- Standard OpenOffice

AEM Forms administrators can create new setting names in addition to these ones.

If you specify conversion settings using the Settings document property, do not specify a value for Conversion Setting Name. If you provide values for both properties, the settings in the Settings document value are used. If you provide no values for either property, the File Type setting that is selected in the default settings on the AEM Forms Server is used. (See Change the default settings in AEM Forms administration help

.)

*Settings document*

(Optional) A `document` value that represents a Generate PDF XML configuration file that contains the file type settings to use. Only the information in the filetype-settings element of the XML is used for this operation.

The XML file must conform to the XML Schema Definition (XSD) for Generate PDF configuration files.

To see an example XML configuration file, use Administration Console to export one. (See Importing and exporting PDF Generator ES2 configuration files in AEM Forms administration help

.)

To provide a literal value, click the ellipsis button and select an asset from an application.

**Output properties**

Property for specifying where to save the optimized PDF document.

*Converted Document*

The location in the process data model to save the optimized PDF. The data type is `document`. For example, to save the optimized PDF in a variable, select a document variable from the list.

**Exceptions**

This operation can throw ConversionException, FileFormatNotSupportedException, and InvalidParameterException exceptions.

### Generate PDF exceptions

The Generate PDF service provides the following exceptions for throwing exception events.

**ConversionException**

Thrown if conversion fails because of an underlying exception.

**FileFormatNotSupportedException**

Thrown when a given file format is not supported.

**InvalidParameterException**

Thrown when a given parameter is not valid.

## 22.31. JDBC

The JDBC service enables processes to interact with databases to accomplish the following tasks:

- Execute stored procedures
- Execute SQL statements
- Query the database

For information about using the JDBC service, see Servicereference.

## JDBC service configuration

The JDBC service can be configured with default properties for connecting to a database server. (See Editing service configurations.)

When you configure the connection property, the JDBC service operations inherit the property values. However, the default connection property can be overridden using the properties of each operation.

### DatasourceName

A `string` value that represents the JNDI name of the data source to use to connect to the database server. The data source must be defined on the application server that hosts the AEM Forms Server. The default is the JNDI name of the data source for the AEM Forms database.

**NOTE:** For WebSphere, use explicit Datasource Name. Do not add any prefix to the WebSphere Datasource Name.

## Call Stored Procedure operation

Executes a stored procedure on the database.

To call stored procedures, the database user account that is used to access the database needs to have the required database permissions.

**NOTE:** Not all databases support stored procedures. See the documentation for the database that you are using for information about whether it supports stored procedures.

For information about the General and Route Evaluation property groups, see Commonoperation properties.

### Input properties

Properties for specifying the stored procedure to call.

*Datasource Name*

A `string` value that represents the JNDI name of the data source to use to connect to the database server. The data source must be defined on the application server that hosts the AEM Forms Server.

**NOTE:** For WebSphere, use explicit Datasource Name. Do not add any prefix to the WebSphere Datasource Name.

*Stored Procedure*

The command to call the stored procedure on the database server. You must use the Callable Statement Info Editor dialog box to create the call command. (See AboutCallable Statement Info Editor.)

### Output properties

Properties for specifying where to store the operation results.

*Number of Rows Affected*

The location to store the number of table rows that the stored procedure affected. The data type is `int`.

**Exceptions**

This operation can throw JDBCConnectionException, JDBCIllegalParameterException, SQLException, and JNDIContextUnavailableException exceptions.

## Execute SQL Statement operation

Executes a SQL statement on a database server and returns the number of rows that were affected. This operation is typically used for SQL statements that do not return a result set, such as INSERT, UPDATE, and DELETE statements.

The database user account that is used to connect to the database server and perform the SQL statement must have the required database permissions.

For information about the General and Route Evaluation property groups, see Commonoperation properties.

**Input properties**

Properties for specifying the stored SQL statement to execute.

*Datasource Name*

A `string` value that represents the JNDI name of the data source to use to connect to the database server. The data source must be defined on the application server that hosts the AEM Forms Server.

**NOTE:** For WebSphere, use explicit Datasource Name. Do not add any prefix to the WebSphere Datasource Name.

*SQL Statement*

A value that represents the SQL statement to execute on the database server. You use the SQL Statement Info Editor to specify the value. (See AboutSQL Statement Info Editor.)

**Output properties**

Properties for saving operation results.

*Number Of Rows Affected*

A location that stores the number of rows that the SQL statement affected. The data type is `int`.

**Exceptions**

This operation can throw JDBCConnectionException, JDBCIllegalParameterException, SQLException, and JNDIContextUnavailableException exceptions.

## Query for Multiple Rows as XML operation

Queries a database using a SQL statement and returns the result set as XML data.

For information about the General and Route Evaluation property groups, see Commonoperation properties.

### Input properties

Properties for defining the query.

*Datasource Name*

A `string` value that represents the JNDI name of the data source to use to connect to the database server. The data source must be defined on the application server that hosts the AEM Forms Server.

**NOTE:** For WebSphere, use explicit Datasource Name. Do not add any prefix to the WebSphere Datasource Name.

*SQL Statement*

The SQL statement to execute on the database server. You must use the SQL Statement Info Editor dialog box to create the SQL statement. (See AboutSQL Statement Info Editor.)

*XML Information*

The definition for the structure of the XML to use for returning the results set. You need to use the XML Document Info Editor dialog box to create the definition. (See AboutXML Document Info Editor.)

### Output properties

Properties for saving operation results.

*XML Document*

The location to use for storing the results set that is returned. The data type is `xml`.

### Exceptions

This operation can throw JDBCConnectionException, JDBCIllegalParameterException, SQLException, and JNDIContextUnavailableException exceptions.

## Query Single Row operation

Queries the database using a SQL statement and saves the first row of the result set.

For information about the General and Route Evaluation property groups, see Commonoperation properties.

### Input properties

Properties for defining the query.

*Datasource Name*

A `string` value that represents the JNDI name of the data source to use to connect to the database server. The data source must be defined on the application server that hosts the AEM Forms Server.

**NOTE:** For WebSphere, use explicit Datasource Name. Do not add any prefix to the WebSphere Datasource Name.

*SQL Statement*

The SQL statement to execute on the database server. You need to use the SQL Statement Info Editor dialog box to create the SQL statement. (See AboutSQL Statement Info Editor.)

### Output properties

Properties for saving operation results.

*Data Mapping*

The locations in the process data model to use for saving the data in the first row of the query result set. Click the ellipsis button to display the SQL Results Mapping Editor dialog box, which you use to specify the data locations.

The dialog box includes a list that you use to associate database columns with process data locations.

**Index:**

An integer value that identifies the column/location pair. The index is one-based, so that the first item in the list has an index value of `1`.

**Column Name:**

The name of the database table column from which data is retrieved using the SQL query. Click the cell and type the column name.

**Process Variable:**

An XPath expression that resolves to the process data location to use to store the data from the corresponding column. Click the cell and then click the ellipsis button to open XPath Builder, which you can use to create the XPath expression.

You can use the Parse Query and Process Metadata buttons to automatically fill the Index and Column Name columns with values:

- Click Parse Query to determine column names from the SQL statement that is specified in the SQL Statement property. You can use this button if the SQL statement explicitly names the column, for example `SELECT column1, column2 FROM table1`. This button does not work for statements such as `SELECT * FROM table1`, or `SELECT concat(a,b) AS FULLNAME FROM table1`.

- Click Process Metadata to retrieve column names from the database server. A request is sent to the database server to process the query and return metadata about the query. The list of columns is determined from the returned metadata.

  After you automatically fill the Index and Column Name columns, you can specify the associated process data locations in the Process Variable column.

  If no value is set for the SQL Statement property, or if neither the Parse Query or the Process Metadata buttons successfully populate the Index and Column Name columns, you need to populate these columns manually. For each column in the result set, click the + button to add a row to the table. After the row is added, you can populate the row with values.

*Number Of Rows Retrieved*

The location in the process data model to use for saving the number of rows in the result set that the query returned. This value is `0` if no rows are in result set and `1` if there is a row in the result set.

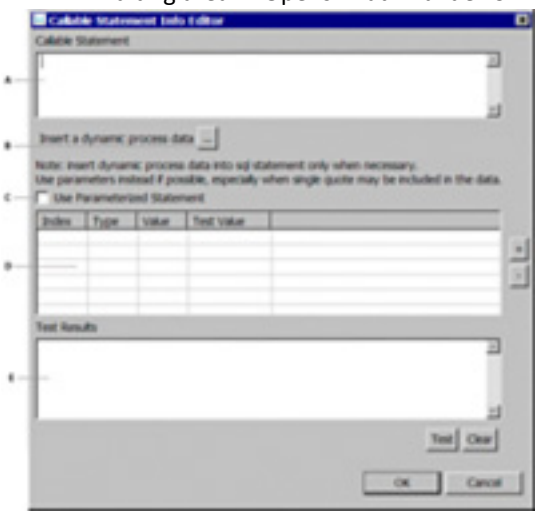
**Exceptions**

This operation can throw JDBCConnectionException, JDBCIllegalParameterException, SQLException, and JNDIContextUnavailableException exceptions.

## About Callable Statement Info Editor

Use the Callable Statement Info Editor dialog box to create and test the call to the database that executes a stored procedure. The call to the stored procedure can include XPath expressions as well as placeholders for parameter values (for stored procedures that take parameter values).

A. Editing area B. Opens XPath Builder C. Enables the use of parameters D. Parameter definitions E. Testing area




You type the procedure call in the Callable Statement box. Procedure calls are in the following format:

```
CALL procedure_name(parameter_list);
```

`procedure_name` is the name of the stored procedure.

`(parameter_list)` is a series of question marks (?) separated by commas. Each question mark represents a parameter.

The number of parameters depends on the procedure that you are calling. If the procedure requires no parameters, this part of the procedure call is omitted.

For example, to call the procedure named `proc1`, which takes two input parameters, the call is `CALL proc1 (?,?);`.

If your procedure call includes placeholders, select Use Parameterized Statement. You must also define the parameter for each placeholder that you include in the statement. (See Definingparameters.) You can also perform the following tasks when creating the statement:

- Include XPath expressions
- Define parametners
- Test

### Include XPath expressions

You can include references to process data in the procedure call using XPath expressions. In the procedure call, XPath expressions must appear inside braces and between dollar signs, as in `{$XPath$}`.

For example, the following procedure call is used for a procedure name that is stored in a process variable named `strvar`:

```
CALL {$/process_data/@strvar$};
```

To create an XPath expression using XPath Builder, click the ellipsis button below the Callable Statement box. The expression is inserted in the statement at the location of the cursor.

### Define parameters

You must define a parameter for each parameter placeholder that the procedure call statement includes. For each parameter placeholder, add a row to the table. The order in which the parameters are defined in the table is the order in which they appear in the procedure call.

For each parameter in the procedure call, click the + button to add a corresponding row to the table, and then specify values in each column.

**Index:**

An index that identifies the parameter. This value is generated automatically when the row is added to the table.

**Type:**

The data type that the parameter holds. Click the cell in the Type column and then select the data type from the list.

**Value:**

The value to use for the parameter. Click the cell in the Value column and either type the value or click the ellipsis button to open XPath Builder to create an XPath expression that resolves to a process data location that holds the value you want to use.

**Test Value:**

A literal value for the parameter to use when testing the procedure call.

**Test**

Click the Test button to test the call to the stored procedure and to see the results.

**NOTE:** Testing executes the stored procedure on the database. Test only in a development environment.

## About SQL Statement Info Editor

Use the SQL Statement Info Editor dialog box to create and test SQL statements that you want to execute on the database server. The SQL statements can include XPath expressions as well as parameter values.

**SQL Statement**

Type the SQL statement in the SQL Statement box. If you want to use a parameter in a SQL statement, use a question mark (`?`) as a placeholder for the parameter. (See Includingparameters.)

**Insert a dynamic process data**

Opens XPath Builder.

**Use Parameterized Query**

Enables the use of parameters.

**Index, Type, Value, Test Value**

Enter parameter definitions.

**Test Results**

Testing area for the statement.

You can also perform the following tasks when creating SQL statements:

- Include XPath expressions
- Define parameters
- Test

**Include XPath expressions**

You can include references to process data in SQL statements using XPath expressions. In SQL statements, XPath expressions must appear inside braces and between dollar signs, as in `{$XPath$}`.

For example, the following SQL statement uses a value stored in the process variable named `first_name` as the condition for retrieving data from `columna` of table1 in the database:

```
SELECT columna FROM table1 WHERE columna LIKE
'{$ /process_data/@first_name$}%'
```

To create an XPath expression using XPath Builder, click the ellipsis button below the SQL Statement box. The expression is inserted in the statement at the location of the cursor.

### Include parameters

To use parameters in SQL statements, place a question mark (?) in the SQL statement to represent the parameters.

For example, the following SQL statement uses a parameter value as the condition for retrieving data from columna of table1 in the database:

```
SELECT columna FROM table1 WHERE columna LIKE (?)
```

For each question mark that you place in the statement, define the parameter in the parameter table. You must also select Use Parameterized Statement.

To define a parameter, add a row to the parameter table. The order in which the parameters are defined in the table is the order in which they appear in the statement. To add a row to the table, click the + button and then specify values in each column.

**Index:**

An index that identifies the parameter. This value is generated automatically when the row is added to the table.

**Type:**

The type of data that the parameter holds. Click the cell in the Type column and then select the data type from the list.

**Value:**

The value to use for the parameter. Click the cell in the Value column and either type the value or click the ellipsis button to open XPath Builder to create an XPath expression that resolves to a process data location that holds the value you want to use.

**Test Value:**

A literal value to use for the parameter when testing the statement. For example, for the statement `SELECT columna FROM table1 WHERE columna LIKE (?)`, the test value could be `A%`.

### Test

Click the Test button to test the SQL statement and to see the results.

**NOTE:** Testing executes the SQL statement on the database. Test only in a development environment.

## About XML Document Info Editor

Use XML Document Info Editor dialog box to define the elements of the XML document that is used to return the results set of a SQL query.

### Root Element Name

A `string` value that represents the name of the root element of the XML document. If no value is specified, `root` is used as the element name.

### Repeating Element Name

A `string` value that represents the name of the XML element to use to contain the information from a row in the result set. The XML document includes one of these elements for each row in the result set. If not value is specified, `element` is used as the element name.

### Column Name Mappings

Use this list to provide the names of the XML elements that store data from database table columns. Each row in the table defines the name of an XML element and the database column that it is associated with.

You use the list to specify values for the following properties.

**Index:**

An integer value that identifies the column/element pair.

**Column Name:**

The name of the database table column that the SQL query retrieves data from. Click the cell and type the column name.

**Element Name:**

(Optional) The name of the XML element to use to contain the data from the table column. Click the cell and type the element name. If you do not specify element names, the column names from the database tables are used as the XML element names.

You can use the Parse Query and Process Metadata buttons to automatically fill the Index and Column Name columns with values:

- Click Parse Query to determine column names from the SQL statement that is specified in the SQL Statement property of the Query For Multiple Rows As XML operation. (See Queryfor Multiple Rows as XML.)

  *You can use this button if the SQL statement explicitly names the column, for example `SELECT column1, column2 FROM table1`. This button does not work for statements such as `SELECT * FROM table1`, or `SELECT concat(a,b) AS FULLNAME FROM table1`.*
- Click Process Metadata to retrieve column names from the database server. A request is sent to the database server to process the query and return metadata about the query. The column names are determined from the returned metadata.

After you automatically fill the Index and Column Name columns, you can specify the associated XML element names in the Element Name column.

If no value is set for the SQL Statement property of the Query For Multiple Rows As XML operation, or if neither the Parse Query or the Process Metadata buttons successfully populate the Index and Column Name columns, you need to populate these columns manually. For each column in the result set, click the + button to add a row to the list. After the row is added, you can populate the row with values.

### Example

The following SQL statement is used for the database query:

```
SELECT column1, column2 FROM table1.
```

No value for Root element Name is specified, so that the name of the root element of the XML is root. No value for Repeating Element Name is specified, so that the name of the element that contains a row of data from the result set is element.

The column names are used as the XML elements that contain the column data. If the query returns a result set with three rows, the XML document has the following structure.

```
<root>
<element>
<column1>row 1 column1 value</column1>
<column2>row 1 column2 value</column2>
</element>
<element>
<column1>row2 column1 value</column1>
<column2>row2 column2 value</column2>
</element>
<element>
<column1>row3 column1 value</column1>
<column2>row3 column2 value</column2>
</element>
</root>
```

## JDBC exceptions

The JDBC service provides the following exceptions for throwing exception events.

### JDBCConnectionException

Thrown when a connection to the database server cannot be established or is lost. The JDBC service configuration may be incorrect, or the database server may be offline.

### JDBCIllegalParameterException

Thrown when the SQL statement contains a parameter that is not supported.

### SQLException

Thrown when an error occurs during execution of the SQL statement.

### JNDIContextUnavailableException

Thrown when the JNDI name of the data source to use to connect to the database server is invalid.

# 22.32. LCCPLM

The Connector for PLM (LCCPLM) service enables processes to interact with a PLM (product life cycle management) system. PLM systems are designed to manage a product through its entire life cycle and integrate people, data, processes, and business systems. For example, PLM data for an engineering project can include documents, images, 2D drawings, and 3D designs.

A process can use the LCCPLM service to retrieve the information to create a PDF document that incorporates all the files associated with a PLM object (for example, an object that represents a specific project). LCCPLM service operations can also save the PDF created using PLM data, and any comments added to it, back into the PLM system. (See "LCCPLM" in Services Reference for AEM Forms

.)

## getDataAndMetaData operation

Retrieves data and metadata for a specified PLM object by using its identifier. Metadata can include information such as a document status or comments.

For information about the General and Route Evaluation property groups, see Commonoperation properties.

### Input properties

Property for specifying the PLM object to retrieve data and metadata for.

*Request parameters*

A `string` value that represents all the needed parameters in XML.

### Output properties

Properties that specify the PLM object data and metadata.

*MasterFile*

(Optional) The file attached to the PLM object. The data type is `document`.

If you provide a variable, from the MasterFile list, select a variable. Click the plus sign button to display the Variable dialog box to create a variable. (See Creatingvariables).

*SecondaryFiles*

(Optional) The list of files associated with the PLM object. The data type is `list`.

If you provide a variable, from the SecondaryFiles list, select a variable. Click the plus sign button to display the Variable dialog box to create a variable. (See Creatingvariables).

*Folder*

(Optional) The folder where files are stored. The data type is `string`.

If you provide a variable, from the Folder list, select a variable. Click the plus sign button to display the Variable dialog box to create a variable. (See Creating variables).

*Doc3d*

(Optional) A `boolean` value that specifies whether the master file is a 3D document.

If you provide a variable, from the Doc3d list, select a variable. Click the plus sign button to display the Variable dialog box to create a variable. (See Creating variables).

*Excluded*

(Optional) A `boolean` value that specifies whether to convert the master file to PDF.

If you provide a variable, from the Excluded list, select a variable. Click the plus sign button to display the Variable dialog box to create a variable. (See Creating variables).

*ProcessName*

(Optional) The name of a AEM Forms subprocess used to create the PDF. The data type is `string`.

If you provide a variable, from the ProcessName list, select a variable. Click the plus sign button to display the Variable dialog box to create a variable. (See Creatingvariables).

*TemplateName*

(Optional) The name of the AEM Forms form template that adds metadata to the PDF. The data type is `string`.

If you provide a variable, from the TemplateName list, select a variable. Click the plus sign button to display the Variable dialog box to create a variable. (See Creatingvariables).

*MetadataMap*

(Optional) The metadata map that contains the form data to merge and the DDX scripts (for example, to add a watermark) to execute. The data type is `map`.

If you provide a variable, from the MetadataMap list, select a variable. Click the plus sign button to display the Variable dialog box to create a variable. (See Creatingvariables).

*ErrorMessage*

(Optional) The description of the error generated when the operation throws an exception. The description includes the message, the class name, and stack trace information. The data type is `string`.

If you provide a variable, from the ErrorMessage list, select a variable. Click the plus sign button to display the Variable dialog box to create a variable. (See Creatingvariables).

*Data and MetaData Descriptor*

(Optional) The data and metadata retrieved by the operation. The data type is `DataAndMetaDataDescriptor`.

If you provide a variable, from the Data and MetaData Descriptor list, select a variable. Click the plus sign button to display the Variable dialog box to create a variable. (See Creatingvariables).

### Exceptions

This operation can throw the LccplmException exception.

## saveCommentsFromDocument operation

Saves comments associated with a PLM object that are captured as PDF data. The comments can be stored in the PLM system with the object.

For information about the General and Route Evaluation property groups, see Commonoperation properties.

### Input properties

Properties for specifying the file that contains the comments to store in the PLM system.

*XDPData*

A `document` value that specifies the file that contains PDF data, including comments.

### Exceptions

This operation can throw the LccplmException exception.

## saveCommentsFromString operation

Saves comments associated with a PLM object that are captured as an XML string. The comments can be stored in the PLM system with the object. Reserved for use with a future release.

For information about the General and Route Evaluation property groups, see Commonoperation properties.

### Input properties

Properties for specifying the comments to store in the PLM system.

*XDPData*

A `string` value that specifies the comments in XFDF (XML Forms Data Format), in Base64 encoding. Adobe Reader 9.2 or later allows you to export PDF content by using XFXF.

### Exceptions

This operation can throw the LccplmException exception.

## storePDF

Checks into the PLM system the PDF that was generated using a PLM object.

For information about the General and Route Evaluation property groups, see Commonoperation properties.

### Input properties

Properties for specifying the PDF to check into the PLM system.

*Request parameters*

A `string` value that represents all the needed parameters in XML.

*path of temporary folder containing the PDF*

A `string` value that represents the folder that contains the PDF and associated files.

*file*

A `document` value that specifies the PDF file to check into the PLM system.

### Output properties

Property to specify error message information for the storePDF operation.

*ErrorMessage*

(Optional) The description to generate when the operation throws an exception. The description includes the message, the class name, and stack trace information. The data type is `string`.

If you provide a variable, from the ErrorMessage list, select a variable. Click the plus sign button to display the Variable dialog box to create a variable. (See Creatingvariables).

### Exceptions

This operation can throw the LccplmException exception.

## LCCPLM exceptions

The LCCPLM service provides the following exception for receiving exception events.

### LccplmException

Thrown when an LCCPLM operation fails. Provides an error code that identifies the error.

# 22.33. LDAP

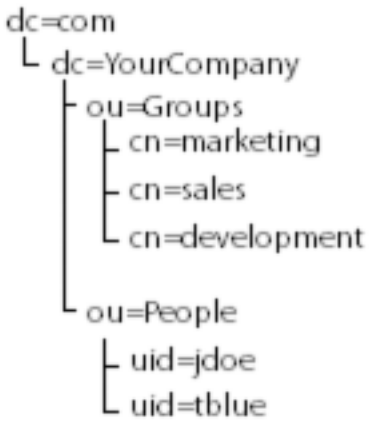
Provides operations for querying LDAP directories. LDAP directories are generally used to store information about the people in an organization. For example, LDAP directories typically store information about the business unit that a person belongs to, information that identifies the person, and information about how to contact them, such as telephone numbers and email addresses.

## Directory structure

LDAP directories use a tree structure as the data model. Different types of databases, such as Sun ONE or Microsoft Active Directory, use different tree structures. LDAP administrators typically customize the directory structure based on the requirements of their organization.

**NOTE:** Talk to your LDAP administrator for information about the directory that you are querying.

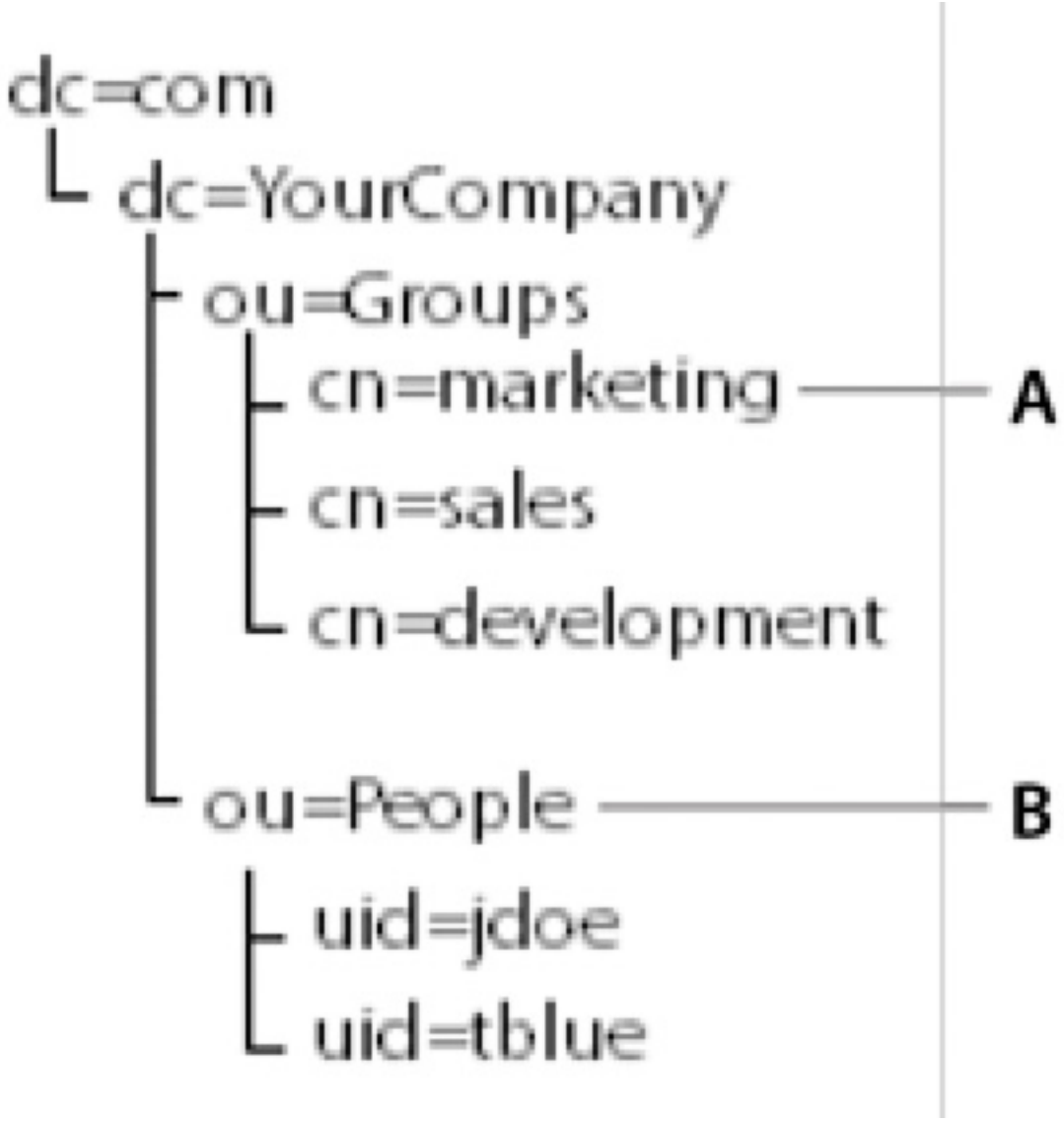
The following graphic shows a very simple directory tree. LDAP directories typically contain many more items, numbering in the thousands.




The structure of the tree and the information that each item in the tree stores is defined by the directory schema.

## Distinguished name

Each item in the tree is uniquely identified by their distinguished name (DN). The DN includes the relative DN of the directory item (for example ou=People) concatenated with the relative DN of the parent items in the tree.




**A.**

The DN of this item is cn=marketing,ou=Groups,dc=YourCompany,dc=com

**B.**

The DN of this item is ou=People,dc=YourCompany,dc=com

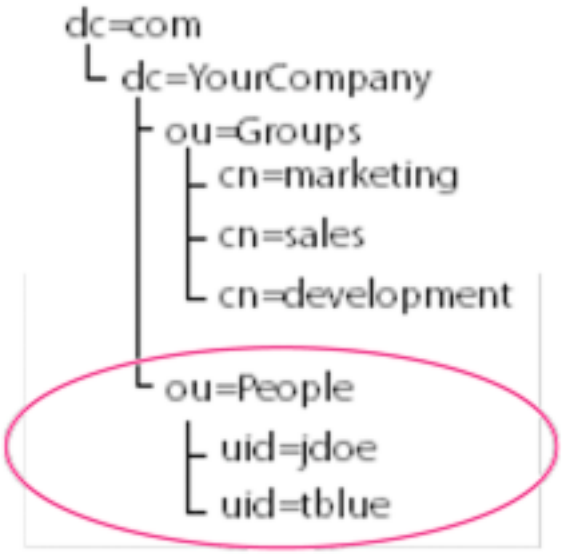
Each item in the directory tree has a number of attributes that are used to store information about the item. For example, items that represent people typically have an attribute named mail which is used to store the person's email address.

The attributes of an item are defined by rules in the directory schema. Each item has the `objectClass` attribute which determines the schema rules that govern the item.

## Base DN

Typically, when you connect to an LDAP server you specify the area of the directory tree that you want to use. To specify the area, you provide the DN of the item in the tree that contains all of the other items that you want to use. This DN is called the base DN.

For example, a base DN of `ou=People,dc=YourCompany,dc=com` provides access to information in that item and the items that it contains.




Using a base DN improves system efficiency because only the information in the base DN is retrieved from the LDAP server.

For information about using the LDAP service, see Services Reference for AEM Forms

.

## LDAP service configuration

Properties used for connecting to the LDAP server.

You must configure these properties before using the operations of the LDAP service. (See Editingservice configurations).

### Initial Context Factory

A `string` value that represents the Java class to use as the context factory. This class is used to create a connection to the LDAP server. The default is `com.sun.jndi.ldap.LdapCtxFactory`, which is appropriate for most LDAP servers.

### Provider URL

A `string` value that represents the URL to use to connect to the LDAP service. The format of the value is `ldap://`*server name*`:`*port*

- *server name* is the name of the computer that hosts the LDAP server
- *port* is the communications port that the LDAP service uses. The default is `389`, which is the standard port used for LDAP connections.

### User Name

A `string` value that represents the user name of the user account to use to log in to the LDAP server. The user account must have permission to connect to the server and read the information in the LDAP directory.

Depending on the LDAP server, the user name could be a simple user name such as “myname” or a DN. For example, “CN=myname,CN=users,DC=myorg”.

### Password

A `string` value that represents the password that corresponds with the user name provided for the Username property.

### Other Properties

A `string` value that represents other properties and their corresponding values that you can provide to the LDAP server. The value is in the following format:

```
property=value;property=value;...
```

## LDAP Query operation

Performs a search on the LDAP server and returns the results which can be saved as process data.

The maximum number of records that the query can return is controlled by the LDAP server. For example, the default configuration of Microsoft Active Directory limits the number of records returned in a result to 1000. If a query is performed on this LDAP server, and more than 1000 records match the query, the result set includes only 1000 records.

**NOTE:** You must configure the LDAP service properties before using the operations of the LDAP service. (See LDAPservice configuration.)

For information about the General and Route Evaluation property groups, see Commonoperation properties.

### LDAP Query Options properties

Properties for specifying details about the query.

*LDAP Query Options*

A value that describes the query to perform, as well as how to save the query results. To specify the value, use the Query LDAP Query Options Editor dialog box. (See AboutLDAP Query Options Editor.)

### Connection Settings properties

Properties for specifying an LDAP connection for the LDAP Query operation. If not provided, or if the Use Global Settings property is selected, the properties specified by the LDAP service configuration are used. (See LDAPservice configuration.)

*Use Global Settings*

A `boolean` value that indicates whether the LDAP connection specified by the LDAP service configuration is used for this operation. If false, the connection specified in the Connection Settings property group is used.

*Initial Context Factory*

A `string` value that represents the Java class to use as the context factory. This class is used to create a connection to the LDAP server. The default is `com.sun.jndi.ldap.LdapCtxFactory`, which is appropriate for most LDAP servers.

*Provider URL*

A `string` value that represents the URL to use to connect to the LDAP service. The format of the value is `ldap://`*server name*`:`*port*

- *server name* is the name of the computer that hosts the LDAP server
- *port* is the communications port that the LDAP service uses. The default is `389`, which is the standard port used for LDAP connections.

*User Name*

A `string` value that represents the user name of the user account to use to log in to the LDAP server. The user account must have permission to connect to the server and read the information in the LDAP directory.

Depending on the LDAP server, the user name could be a simple user name such as “myname” or a DN. For example, “CN=myname,CN=users,DC=myorg”.

*Password*

A `string` value that represents the password that corresponds with the user name provided for the Username property.

*Other Properties*

A `string` value that represents other properties and their corresponding values that you can provide to the LDAP server. The value is in the following format:

```
property=value;property=value;...
```

### Exceptions

This operation can throw LDAPConnectionException and LDAPQueryException exceptions.

## LDAP Query to XML operation

Performs a search on the LDAP server and returns the results in an XML document, which you can save as process data.

The maximum number of records that the query can return is controlled by the LDAP server. For example, the default configuration of Microsoft Active Directory limits the number of records returned in a result to 1000. If a query is performed on this LDAP server, and more than 1000 records match the query, the result set includes only 1000 records.

**NOTE:** You must configure the LDAP service properties before using the operations of the LDAP service. (See LDAPservice configuration.)

For information about the General and Route Evaluation property groups, see Commonoperation properties.

### LDAP and XML Options properties

Properties for specifying details about the query.

*LDAP and XML Options*

A value that describes the query to perform and how to format the XML document that contains the query results. To specify the value, use the LDAP Query Options Editor dialog box. (See AboutLDAP Query Options Editor.)

### Connection Settings properties

Properties for specifying an LDAP connection for the LDAP Query to XML operation. If not provided, or if the Use Global Settings property is selected, the properties specified by the LDAP service configuration are used. (See LDAP serviceconfiguration.)

*Use Global Settings*

A `boolean` value that indicates whether the LDAP connection specified by the LDAP service configuration is used for this operation. If false, the connection specified in the Connection Settings property group is used.

*Initial Context Factory*

A `string` value that represents the Java class to use as the context factory. This class is used to create a connection to the LDAP server. The default is `com.sun.jndi.ldap.LdapCtxFactory`, which is appropriate for most LDAP servers.

*Provider URL*

A `string` value that represents the URL to use to connect to the LDAP service. The format of the value is `ldap://`*server name*`:`*port*

- *server name* is the name of the computer that hosts the LDAP server

- *port* is the communications port that the LDAP service uses. The default is `389`, which is the standard port used for LDAP connections.

*User Name*

A `string` value that represents the user name of the user account to use to log in to the LDAP server. The user account must have permission to connect to the server and read the information in the LDAP directory.

Depending on the LDAP server, the user name could be a simple user name such as “myname” or a DN. For example, “CN=myname,CN=users,DC=myorg”.

*Password*

A `string` value that represents the password that corresponds with the user name provided for the Username property.

*Other Properties*

A `string` value that represents other properties and their corresponding values that you can provide to the LDAP server. The value is in the following format:

```
property=value;property=value;...
```

**Output XML Document properties**

Properties for saving the query results.

*Output XML Document*

The location to store the XML document that contains the query results. The data type is `xml`.

**Exceptions**

This operation can throw LDAPConnectionException, LDAPQueryException , and LDAPDOMException exceptions.

## LDAP exceptions

The LDAP service provides the following exceptions for receiving exception events.

**LDAPConnectionException**

Thrown when a connection to an LDAP server cannot be established or is lost. The LDAP service configuration may be incorrect or the server may be offline.

**LDAPQueryException**

Thrown when an error occurs during execution of the LDAP query.

### LDAPDOMException

Thrown when an error occurs during the creation of the XML document that contains the query results.

## LDAP Query Options Editor

The LDAP Query Options Editor dialog box enables you to specify details about LDAP queries for the LDAP Query and LDAP Query To XML operations. The dialog box provides several tabs that you use to specify the query to perform and how to return the query results. The service operation that you use determines which tabs appear in the dialog box.

If the LDAP service configuration has been performed, the dialog box connects to the LDAP server when you open it. (See LDAPservice configuration.) The connection enables the dialog box to populate the properties on the tabs with values that are based on the LDAP directory configuration.

### Query tab

Use this tab to specify the query to perform. The values that you specify for each property of the query determine which directory items (or object types) are returned by the query.

*Base-DN*

The item at the top level of the LDAP directory tree that includes the directory branches to make available to your queries. Specifying a base DN can increase the efficiency of queries when the LDAP directory has many top-level objects.

By default, the root DN is the base DN. Use the Base-DN list to select a different node at the top level. Click Load to ensure that the list is populated with current information from the LDAP server.

*Search Context*

A `string` value that represents the starting point for searches in the LDAP directory tree. The search is conducted in the branches of the LDAP directory tree that are contained in the directory item that you specify.

You can type the DN of the starting point or use an XPath expression if the value is saved as process data. The XPath expression must be enclosed within braces and dollar signs:

```
{$expression$}
```

*Nested Context*

A `string` value that represents the attribute that contains the DN of another directory item. This directory item is retrieved in the query results instead of the items found based on the query's search context.

For example, the search context matches directory entries that represent employees. The value of the `manager` attribute of the returned entries is the DN of another directory entry, which represents the employee's manager. Specifying `manager` for the value of the Nested Context property returns the directory entry for the employee's manager.

The directory entry for the manager also includes the `manager` attribute. Specifying `manager/manager` for the value of the Nested Context property returns the directory entry for the manager's manager.

*Search Filter*

A `string` value that represents the search criteria for the query. The query returns information from the directory items that contain attribute values that match those described in the search filter. (See Searchfilter syntax.)

To specify a search filter, use the Search Query Builder dialog box (see About SearchQuery Builder) and click the ellipsis button to display the dialog box.

*Search Scope*

A `string` value that represents the number of levels of information to search relative to the search context. The following values are valid.

**OBJECT:**

Only the directory item specified for the Search Context property is searched. This is the smallest scope.

**ONE LEVEL:**

The level below the search context searched, as well as the directory item specified for the Search Context property.

**SUBTREE:**

All levels below the search context are searched, as well as the directory item specified for the Search Context property. This is the largest scope.

You should use the smallest scope that is necessary for the purposes of your query. The scope determines the number of directory items that are compared to the search filter. More server resources are used and more time is required to compare larger numbers of records.

**Output (LDAP Query) tab**

Use this tab to specify the information that you want to save from the search results. You save the information from specific attributes of the directory items. The values are saved in locations in the process data model.

*If Multiple Objects Are Returned*

Specifies whether you want to save information from only the first directory item that is returned by the query, or whether you want to save information from all of the returned items. If you want to save information from all items, you also specify the format of the retrieved data.

**Use First:**

Saves information only from the first item that is returned. The data type is determined by the data you are retrieving.

**Concatenated As String:**

For each directory item that is returned, the value of an attribute for each item is concatenated into a single `string` value. The character used to delimit each attribute value is specified in the Object Delimiter property on the Output Settings tab.

**List:**

For each item that is returned, the attribute value of each record is stored as a `string` value in a `list` value.

*Save Result Count Into*

The location to save the number of items that matched the search filter. The data type is `int`. If you save information from only the first returned directory item, this value enables you to determine if there are additional matching items found. It also enables you to determine if there are no items found.

*Storage locations*

The attributes of the returned directory items that you want to save, and locations in which to save the values. Click the + button to add a row to the table. After the row is added, you can populate the row with values.

**Value:**

The name of the attribute that you want to save. Each cell in the Value column is a menu from which you can select an attribute name.

**XPath:**

The location to store the attribute value. The data type is determined by the value that you specified for the If Multiple Objects Are Returned property. Each cell in the XPath column provides an ellipsis button that opens XPath Builder which you can use to provide an XPath expression for the data location.

**Output (LDAP Query To XML) tab**

Use this tab to specify the names of XML elements to use for storing retrieved attribute values. The XML document has the following structure.

```
<Root_Element>
<Repeated_Element>
<element1>attribute value</element1>
<element2>attribute value</element2>
<element3>attribute value</element3>
.
.
.
```

```
</Repeated_Element>
<Repeated_Element>
<element1>attribute value</element1>
<element2>attribute value</element2>
<element3>attribute value</element3>
.
.
.
</Repeated_Element>
.
.
.
</Root_Element>
```

*Root Element*

A `string` value that represents the name of the root element of the XML document.

*Repeated Element*

A `string` value that represents the name of the XML element to use to contain the information from a directory item. One of these elements exists for each directory item that the search returns.

*Attribute/Element list*

A list of directory item attributes and the names of the corresponding XML elements to use to store the attribute values. Click the + button to add a row to the table. After the row is added, you can populate the row with values.

**Attribute:**

The name of the attribute for the found directory item or items that you want to save. Each cell in the Attribute column is a menu from which you can select an attribute name.

**Element:**

The name of the XML element to use to contain the attribute value. Each cell in the Element column provides an ellipsis button that opens XPath Builder which you can use if the element names are saved in the process data model. If no value is specified, the name of the attribute is used as the element name.

**Output Settings (LDAP Query) tab**

This tab enables you to specify options for formatting the data that is retrieved from the LDAP directory.

*Nested Context*

A `string` value that represents the attribute that contains the DN of another directory item. This directory item is retrieved in the query results, in addition to the items found based on the query's search context.

For example, the search context matches directory entries that represent employees. The value of the `manager` attribute of the retrieved entries is the DN of another directory entry, which represents the employee's manager. Specifying `manager` for the value of the Nested Context property retrieves the directory entry for the employee's manager.

The directory entry for the manager also includes the `manager` attribute. Specifying `manager/manager` for the value of the Nested Context property retrieves the directory entry for the manager's manager.

*Attribute Delimiter*

A `string` value that represents the delimiter to use for values within attribute values. When attribute values are comprised of multiple values, these values are separated by the attribute delimiter.

*Attribute Escape Delimiter*

A `string` value that represents the character to use to escape the delimiters that are used in attribute values, when the values are comprised of multiple values.

*Object Delimiter*

A `string` value that represents the delimiter for objects that are returned when Concatenated as String is selected for the If Multiple Objects Are Returned property.

For example, a query returns two objects, `John` and `Mary`. `John` was a member of the groups `groupA` and `groupB`. `Mary` is a member of the three groups `groupX`, `groupY`, and `groupZ`. The object delimiter is; and the attribute delimiter is `$$`. The attributes contain the following strings:

```
name: John$$Mary
memberOf: groupA;groupB$$groupX;groupY;groupZ
```

*Object Escape Delimiter*

A `string` value that represents the character to use to escape characters in object names that happen to be the same as the delimiters that are used to separate objects.

**Test tab**

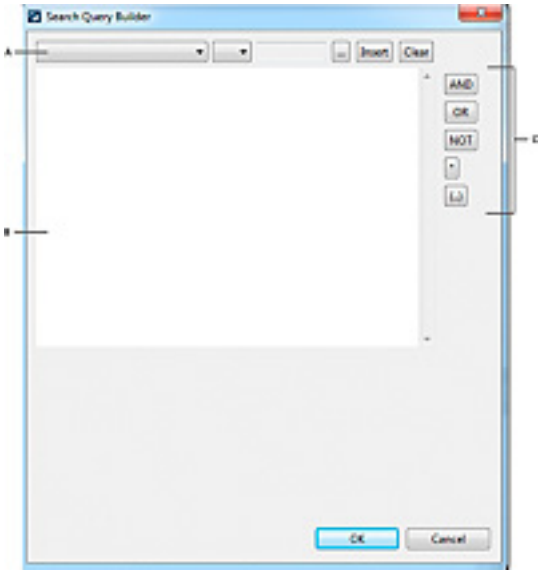
This tab enables you to test the query. Click Test to see the results of the query.

**NOTE:** To perform the test, all property values must be specified as literals because XPath expressions can not be evaluated at design time.

The test only shows the data that is retrieved from the LDAP server, and does not show how the results are saved.

## Search Query Builder

You can use the Search Query Builder dialog box for creating search filters. The Search Query Builder provides a list of directory item attributes that you can use in the search filter and features for inserting special characters defined by search filter syntax. (See Searchfilter syntax.)




**A.**

Tools for specifying attribute conditions

**B.**

Editing area

**C.**

Tools for inserting syntactical elements

### Using the editing area

The editing area contains the text used as the search filter. You can either type the search filter directly in the editing area, or you can use the other tools that Search Query Builder provides.

### Referencing process data

You can make references to process data in search filters using XPath expressions. In the search filter, XPath expressions must appear inside braces and between dollar signs, as in `{$expression$}`. The following search filter includes the XPath expression `/process_data/@stringvar`:

```
(uid={$/process_data/@stringvar$})
```

### Specifying attribute conditions

Use the tools along the top of the Search Query Builder dialog box for creating and adding basic search filter expressions that specify attribute conditions, for example `(mail=jdoe@adobe.com)`. The tools enable you to select the attribute upon which you want to apply a condition. You can also specify the arithmetic operator that relates the attribute to a value. You can open XPath Builder to retrieve a value that is stored as process data.

**A.**

Use to select the attribute upon which you want to apply a condition.

**B.**

Use to select the arithmetic operator to use in the expression.

**C.**

Displays the data location from the process data model.

**D.**

Use to select a location from the process data model.

**E.**

Inserts the filter expression into the editing area.

**F.**

Clears the text from the editing area.

After you select the attribute name and operator from the list, you can specify the value to compare to the attribute using one of the following methods:

- Use XPath Builder to specify a location in the process data model where the value is stored, and then press Insert.
- Press Insert to add the attribute name and operator to the editing area, and type the value directly into the editing area.

  *The following text is inserted when the* `uid` *attribute, the* `=` *operator, and the XPath expression that resolves to a process variable named stringvar is selected:*

  ```
  uid={$/process_data/@stringvar$}
  ```

  **NOTE:** To make the above expression a valid search filter, enclose it in parentheses.

### Inserting syntactical elements

Use the tools along the right side of the Search Query Builder dialog box for inserting syntactical elements into the search filter.

The button label denotes the text that the button inserts. For example, the OR button inserts `OR`. The text is inserted at the location of the cursor in the editing area.

## Search filter syntax

LDAP search filters define criteria for selecting items from a directory. The criteria are based on attribute values. The syntax for search filters is defined in RFC2254 (The String Representation of LDAP Search Filters).

The simplest filter places a condition on a single attribute value:

```
(attributeType filterType value)
```

- Filters must be within parentheses.

- `attributeType` is the name of the attribute upon which you are placing the condition.
- `filterType` is one of four valid arithmetic operators.
- `value` is the value that you are comparing to the attribute.

The following table lists the valid operators that you can use in a search filter.

| Operator | Meaning |
|---|---|
| = | equal |
| ~= | approximately equal |
| <= | less than or equal to |
| >= | greater than or equal to |

For example, the search filter `(uid=jdoe)` returns the directory item that has the uid attribute of value `jdoe`.

### Substrings and any values

In search filters, the asterisk (`*`) represents any sequence of characters. You can use the asterisk for expressing values that have specific prefixes or suffix, or to express any value.

- The expression `(uid=j*)` matches all items with a `uid` attribute that begins with `j`.
- The expression `(uid=*doe)` matches all items with a `uid` attribute that ends with `doe`.
- The expression `(uid=*)` matches all items that have a `uid` attribute of any value.

### Logical operators

Use logical operators to apply conditions on more than one attribute, or to apply the opposite of the condition specified by a filter. Logical operators precede the filters to which they are applied. The following table lists the logical operators and provides examples of their use.

| Logical operator | Description | Example |
|---|---|---|
| `&` | All associated filters match. | `(&(uid=j*)(c=CA))`<br>Matches all directory items that have a `uid` attribute value that begins with `j` and a `c` attribute value that equals `CA`. |
| `\|` | Any of the associated filters match. | `(\|(c=CA)(c=US))`<br>Matches all directory items that have a `c`attribute value that equals either `CA` or `US`. |
| `!` | The opposite of the filter. | `(!(uid=j*))`<br>Matches all directory items that have a `uid` attribute value that does not begin with `j`. |

### Escape character

To express the literal value of a special character, precede the character with a backslash (\). For example, if an attribute value includes parentheses, precede the opening and closing parenthesis with the backslash:

```
(telephoneNumber=\(555\) 555-1234)
```

### All directory items

All directory items must have a value for the objectClass attribute. The following search filter matches all items in the area of the directory that is searched:

```
(objecClass=*)
```

## 22.34. Output

The Output service merges data with a form design, created in Designer, and generates the output in various formats. Data is provided as a series of records. Records represent an XML element that stores a set of form data values for merging with a form design. You can have multiple records to represent multiple groups of related data to merge with a form design. The output is created as an output stream that can be sent to a network printer, local printer, email attachment, or saved as a PDF document. The service is useful for processes that merge large sets of data to generate output, which is typical when providing on-demand document-generation services.

For example, use the generatePrintedOutput operation to merge ten records with a form design to generate an output format for each record. The form design is saved in an application to maximize portability. The records are located within an XML data source. Then, using the sendToPrinter operation, you can send each output to a printer.

**NOTE:** If you process large batches of records or input files, ensure that the Default Document Disposal Timeout value in administration console is set high enough. In administration console, select Settings > Core System Settings > Configuration. (See Core System SettingsHelp
.)

When you merge a form design with data, the Output service can generate these output formats:

- A PDF or PDF/A document output as PDF files.
- Adobe PostScript
- Printer Control Language (PCL)
- Zebra Programming Language (ZPL)
- Datamax Printer Language (DPL)
- Intermec Printer Language (IPL)
- TEC Printer Command Language (TPCL)

**TIP:** Use the OutputIVS sample to test your form designs. Testing the form design is useful for troubleshooting the form design and for accessing the generated output.

The Output service can send output to a printer by using common printer protocols. You can use default XDC files to specify printer settings or create custom XDC files in XDC Editor. The form designs, data files,

and XDC files that are used can be stored in various locations to meet your requirements. The locations include an application, the repository in AEM Forms Server, the HTTP location, a network location, or the local file system.

In addition, you can flatten an interactive PDF form. The process of *flattening*a PDF document converts an interactive PDF document to a non-interactive PDF document. When you flatten a PDF document, users cannot modify form fields in the PDF document. Typically, you flatten a PDF document for archiving or printing. However, signature fields on the PDF can remain interactive, depending on the operation you use and how you configure properties in the operation.

**TIP:** To improve performance, use Output operations in short-lived processes when you use Open Type fonts for PCL. See AEM Forms XDC EditorHelp

.

For information about using the Output service, see Services Referencefor AEM Forms

.

## Retaining legacy appearance of PDF Forms in AEM Forms

In AEM Forms, PDF forms rendered with the assistance of Forms, Output, or Form Data Integration (FDI) services have the appearance that is similar to PDF forms in Acrobat X. However, if you are upgrading from the previous version of AEM Forms, and would like to retain the older appearance of PDF forms, you can set a service-level configuration option for the Forms, Output, and FDI services.

For the Forms, Output, or FDI services, when the value of this option (Appearance Compatibility Mode) is set to 9, all PDF forms rendered using the service appear in the legacy appearance. Any other value set for this option results in generated PDF forms appearing with the look and feel enabled by AEM Forms.

1) In the Components view, drill down to **Output** > **Active Services** > **OutputService:1.1**.
2) Right click OutputService:1.1 and select Edit Service Configuration.
3) On the Configuration Settings dialog, edit the Appearance Compatibility Mode option to set the version of Acrobat whose appearance mode is to be configured.
4) Click OK to complete the setting.

## generatePDFOutput operation

Generates a PDF or a PDF/A document from the provided form design and data. Optionally, generate a metadata file for each record and send the output directly to a printer. Use this operation to use a form design that is part of an application in the Applications view. It is recommended that you use the generatePDFOutput operation when the form design is located on a network, local file system, or HTTP location.

The benefits of using this operation instead of the generatePDFOutput (deprecated) operation are as follows:

- Maximizes application portability because all assets are contained in a single application.
- Simplifies the process design because the form design or data files can be assessed as a `document` variable.

For example, your application must create a PDF file for hundreds of records in a data file. The form design is part of an application. Use the generatePDFOutput operation to create a PDF file for each record in the data file.

**NOTE:** Use the generatePDFOutput(deprecated) operation only when you cannot create a document variable to store the form design or form data. You cannot create document variables for form designs or data files that are dynamically referenced on a network, local file system, or HTTP location.

For information about the General and Route Evaluation property groups, see Commonoperation properties.

### Input properties

Properties to specify a form design, data file, and print format.

*Form*

A `document` value that represents the form design. You create form designs in Designer.

If you provide a literal value, clicking the ellipsis button opens the Select Asset dialog box. (See About-Select Asset.)

*Input Data*

(Optional) A `document` value that specifies the data file that is merged with the form design. The data file that you provide is an XML file.

If you provide a literal value, clicking the ellipsis button opens the Select Asset dialog box. (See AboutSelect Asset.)

*Transformation Format*

A `TransformationFormat` value that specifies the format that the document is rendered to.

If you provide a literal value, select one of these values:

**PDF:**

A non-interactive PDF document is created.

**PDFA:**

A non-interactive PDF/A document is created. PDF/A documents are used for archiving purposes and based on ISO standard 19005-1. It also embeds all the fonts and turns off compression. If you select this value, provide values for the PDF/A Revision Number and PDF/A Conformance properties.

### Template Options properties

*Content Root*

(Optional) A `string` value that specifies the URI, absolute reference, or location in the repository to retrieve relative assets used by the form design. For example, if the form design references an image

relatively, such as ../myImage.gif, myImage.gif must be located at repository://. The default value is repository://, which points to the root level of the repository.

When you pick an asset from your application, the Content Root URI path must have the correct structure. For example, if a form is picked from an application named SampleApp, and is placed at SampleApp/1.0/forms/Test.xdp, the Content Root URI must be specified as `repository://administrator@password/Applications/SampleApp/1.0/forms/`, or `repository:/Applications/SampleApp/1.0/forms/` (when authority is null). When the Content Root URI is specified this way, the paths of all of the referenced assets in the form will be resolved against this URI.

Use the following sources for a URI or absolute reference:

**Repository:**

The repository contains assets that you upload to the AEM Forms Server. The value `repository:///` references the root of the repository. The first two forward slashes are part of the protocol (`repository://`) and the third forward slash represents the root of the repository. For example, if the form design references an image relatively, such as ../myImage.gif, myImage.gif must be located at repository://. The default value is repository://, which points to the root level of the repository.

**Directory in the file system of the AEM Forms Server:**

You can specify a location on the AEM Forms Server, such as `C:\`*[folder name]*. Using a location on the server is not recommended when maximizing the portability of an application.

**Network directory:**

You can specify a location on the network, such as `\\[folder name.]`

**Web location that is accessible by using HTTP:**

After you upload a file to a location on a web server, you can specify the location by using a URL. For example, type `http://`*[server name]*`:`*[port number]*`/`*[folder name]*. The value *[server name]* is the name of the web server, *[port number]* is the port number, and *[folder name]* is the name of the folder.

**PDF Output Options properties**

Properties for specifying output options for the generated PDF file.

*XCI URI*

(Optional) A `string` value that specifies the XCI file to use. XCI files are used to describe fonts that are used for form design elements. You can also specify print options, such as number of copies, whether duplex printer is used, or stapler options.

*Character Set*

(Optional) A `string` value that specifies the character set used to encode the rendered form.

If you provide a literal value, select the character set to use or select one of these values:

**<Use Server Default>:**

(Default) Use the Character Set setting that is configured on the AEM Forms Server. The Character Set setting is configured using administration console. (See Forms administrationhelp

.)

**<Use Custom Value>:**

Use a character set that is not available in the list. After you select this value, in the box beside the list, type the canonical name (Java.nio API) of the encoding set that is not in the list. For a list of character sets, see http://java.sun.com/j2se/1.5.0/docs/guide/intl/encoding.doc.html.

*Locale*

(Optional) A `string` value that specifies the language used for generating the PDF document.

If you provide a literal value, select a language from the list or select one of these values:

**<Use Server Default>:**

(Default) Use the Locale setting that is configured on the AEM Forms Server. The Locale setting is configured using administration console. (See Forms administrationhelp

.)

**<Use Custom Value>:**

Use a locale that is not available in the list. After you select this value, in the box beside the list, type the Locale ID of the locale code that is not in the list. For a list of supported locale codes, see http://java.sun.com/j2se/1.5.0/docs/guide/intl/locale.doc.html.

*Record Name*

(Optional) A `string` value that specifies the element name that identifies the beginning of a batch of records.

*Record Level*

(Optional) An int value that specifies the XML element level that contains the record data.

If you provide a literal value, the default is 1, which represents the first level in the record specified by the Record Name property.

*Generate Multiple Streams*

(Optional) A `boolean` value that specifies whether the operation creates a single output or multiple outputs.

If you provide a literal value, the Generate Multiple Streams check box is deselected by default. Select the check box to create multiple output. Multiple outputs are useful for sending each record after it is completed processing. Deselect the check box to create a single output. Single output is useful when you want to send all the processed records at once.

*Enable Lazy Loading*

(Optional) A `boolean` value that specifies whether incremental (lazy) loading is used when processing multi-record data sets. When incremental loading is used, it helps to reduce the amount of memory used on the AEM Forms Server. The use of incremental loading limits XLST options specified in the XCI file because transformations can only be applied to one record at a time.

If you provide a literal value, select the Enable Lazy Loading check box to load and merge one record of data. Deselect the check box to load and merge the entire data file, or all data at one time.

*Pattern Match Size*

(Optional) An int value that specifies the number of bytes to use from the beginning of the input data file to scan for the pattern strings.

If you provide a literal value, the default is 500.

*Pattern Matching Rules*

(Optional) A `list` of `string` values for scanning the data file for a pattern and associating the data with a specific form design. Click one of these buttons to add or delete an entry from the list.

**Add A List Entry:**

Adds a new rule. After you click this button, a new entry is created in the list. In the Pattern field for the new entry, type a pattern to search for. In the Form field, type the name of the form design for the matching pattern. All the form designs that you specify must be available at the location specified by the Content Root property in this operation.

**Delete A Selected List Entry:**

Removes the selected rule from the list.

For example, you can to search for a text pattern such as *car*, and specify the operation to use the form design named *AutoInsurance.xdp*. The form AutoInsurance.xdp is used whenever the text *car* is in the data. For information about working with search rules, see Designer Help

.

*Output Location URI*

(Optional) A `string` value that specifies the URI to save the output file. If you create multiple files, the filenames are suffixed with a numeric value. For example, if you specify C:\forms\Loan.pdf, the Output service creates the filenames Loan0001.pdf, Loan0002, Loan0003, and so on. Each of the files are created in the C:\forms folder on the AEM Forms Server.

*Printer Name*

(Optional) A`string` value that specifies the name of the printer for sending the output for printing. The AEM Forms Server must be configured to access specified printer.

If you provide a literal value, you can specify the name of printer in a format as *[print server]**[printer name]*. The value *[print server]* represents a printer server and *[printer name]* is the name of the printer.

*LPD URI*

(Optional) A`string` value that specifies the URI of the Line Printer Daemon (LPD) running on the network.

If you provide a literal value, you can specify the name of the LPD in a format as `lpd://`*[host name]*. The value *[host name]* is the name of the LPD host.

*LPD Printer Name*

(Optional) A`string` value that specifies the name of the printer on the specified Line Printer Daemon (LPD).

If you provide a literal value, you can specify a printer name in a format as *[printer name]*.The value *[printer name]* is the name of the printer.

*Meta Data Spec File*

(Optional) A `string` value that specifies the URI of the metadata spec file to use. A metadata spec file is used to generate metadata from the provided data file.

For example, a metadata spec file can be created as follows:

```
<metadata-spec>
<system>
<map xpath="$fileName" name="fName"/>
<map xpath="$format" name="docType"/>
</system>
<user>
<map xpath="header/txtOrderedByCompanyName" name="companyName"/>
<map xpath="header/txtOrderedByAddress" name="address"/>
<map xpath="header/txtOrderedByCity" name="city"/>
<map xpath="header/txtOrderedByStateProv" name="state"/>
<map xpath="header/txtOrderedByZipCode" name="zipCode"/>
<map xpath="header/txtOrderedByCountry" name="country"/>
<map xpath="header/txtOrderedByPhone" name="phone"/>
<map xpath="header/txtOrderedByFax" name="fax"/>
</user>
</metadata-spec>
```

If you provide a literal value, clicking the ellipsis button opens the Select Asset dialog box. (See AboutSelect Asset.)

A metadata file is created with the data that is extracted from each record. The structure of the metadata file that is created is based on the metadata spec file. The resulting metadata file that is based on the previous metadata spec file is as follows:

```
<root><record id='null'>
<system>
<data name='fName'>c:\cumulativedata0001.pdf</data>
<data name='docType'>PDF</data>
</system>
<user>
```

```
<data name='companyName'>Any Company Name</data>
<data name='address'>555, Any Blvd.</data>
<data name='city'>Any City</data>
<data name='state'>Alabama</data>
<data name='zipCode'>12345</data>
<data name='country'>United States</data>
<data name='phone'>(123) 456-7890</data>
<data name='fax'>(123) 456-7899</data>
</user>
</record>
```

*Record ID XPath*

(Optional) An int value that specifies the root level node to use for XPath expressions. The root level node specifies the starting point for XPath expressions. For example, consider a data schema as follows:

```
<batch_100>
<purchaseOrder>
<header>
<txtPONum>1of100</txtPONum>
<dtmDate>2004-02-08</dtmDate>
<txtOrderedByCompanyName>Any Company Name</txtOrderedByCompanyName>
<txtOrderedByAddress>555, Any Blvd.</txtOrderedByAddress>
<txtOrderedByCity>Any City</txtOrderedByCity>
<txtOrderedByStateProv>Alabama</txtOrderedByStateProv>
</header>
</detail
<txtPartNum>580463116</txtPartNum>
<txtDescription>Electric Fuel Pump</txtDescription>
<numQty>1</numQty>
<numUnitPrice>149.95</numUnitPrice>
<numAmount>149.95</numAmount>
</detail>
</purchaseOrder>
</batch_100>
```

To reference the `<purchaseOrder>` tag as the first level in your XPath expression, set this property to value of 2. To set `<batch_100>` as the first level, set this property to the value of 1.

*Generate Record Level Meta Data*

(Optional) A `boolean` value that specifies whether to generate a metadata file for each record.

If you provide a literal value, the Generate Record Level Meta Data check box is deselected by default. Select the check box to generate a metadata file for each record that is processed. Deselect the check box to generate one metadata file for all records that are processed.

**General Render Options properties**

Properties for specifying basic PDF rendering options.

*Linearized PDF*

(Optional) A `boolean` value that specifies whether to render a linearized PDF document. A linearized PDF document is organized so that it supports incremental access to the PDF document when accessed on the network. For example, a linearized PDF can be displayed in a web browser before the entire PDF document is downloaded.

If you provide a literal value, the Create PDF For Quicker Display In A Browser check box is deselected by default. Select the check box to render a linearized PDF form. This option is best used for optimized web applications. Deselect the check box to not render a linearized PDF form. This option is best used for non-web applications.

*Debug Enabled*

(Optional) A `boolean` value that specifies whether debug-level logging is turned on. Debug-level uses more server resources and can negatively affect performance on the AEM forms Server. The default value is False. Debug-level logging provides more information in the J2EE application server's log file for debugging.

If you provide a literal value, the Debug Enabled check box is deselected. Select the check box to enable debug-level logging. Deselect the check box to use default-level logging.

**PDF Specific Render Options properties**

Properties for specifying advanced options for rendering the PDF file.

*Acrobat Version*

(Optional) An `AcrobatVersion-OutputService` value that specifies the minimum Acrobat and Adobe Reader version that is required to view the PDF document. Any version later than the specified version can also open the PDF document. In addition, this property specifies the PDF version that is used to generate the PDF document.

If you provide a literal value, elect one of these values:

**<Use Form Template Default>:**

(Default) The Target Version setting in the form design determines the minimum the version of Acrobat or Adobe Reader. In addition, the form design determines the PDF version.

**Acrobat and Adobe Reader 6 or later:**

PDF Version 1.5 is used to generate the PDF document.

**Acrobat and Adobe Reader 7.0 or later:**

PDF Version 1.6 is used to generate the PDF document.

**Acrobat and Adobe Reader 7.0.5 or later:**

PDF Version 1.65 is used to generate the PDF document.

**Acrobat and Adobe Reader 8 or later:**

PDF Version 1.7 is used to generate the PDF document.

**Acrobat and Adobe Reader 8.1 or later:**

PDF Version 1.7-ADBE-1 is used to generate the PDF document.

**Acrobat and Adobe Reader 9 or later:**

PDF Version 1.7-ADBE-3 is used to generate the PDF document.

**Acrobat and Adobe Reader X:**

PDF Version 1.7-ADBE-3 is used to generate the PDF document.

*Accessible (Tagged) PDF*

(Optional) A `boolean` value that specifies whether to create a tagged Adobe PDF form. A *tagged PDF form* defines a set of standard structure types and attributes that support the extraction of page content and reuse for other purposes. It is intended for use by client applications that perform the following types of operations:

- Simple extraction of text and graphics for pasting into other applications
- Automatic reflow of text and associated graphics to fit a page of a different size than was assumed for the original layout
- Processing text for such purposes as searching, indexing, and spell-checking
- Conversion to other common file formats (such as HTML, XML, and RTF) with document structure and basic styling information preserved
- Making content accessible by screen reader software

If you provide a literal value, the Create a PDF That Meets The Section 508 Accessibility Rules check box is deselected by default. Select the check box to render a tagged PDF form. This value is not valid for PDF/A output because PDF/A 1A is always tagged and PDF/A 1B is never tagged. Deselect the check box to create an inaccessible form.

*Render At Client:*

(Optional) A `string` value that specifies whether to enable the delivery of PDF content by using the client-side rendering. Client-side rendering is available in Acrobat 7.0 or Adobe Reader 7.0 and later. Valid string values you can provide are auto, yes, and no. Use client-side rendering to improve the performance of the Output service.

If you provide a literal value, <Use Form Template Default> is selected by default. Select one of these values:

**<Use Form Template Default>:**

The Output service determines the form rendition based on the setting in the form design.

**Yes:**

A dynamic PDF form is generated and rendering occurs in Acrobat. Rendering of a dynamic form occurs only for Acrobat 7.0 or later. For earlier version of Acrobat, no rendering occurs.

**No:**

A static PDF form is generated. No rendering occurs on the client.

*Retain Signature Field:*

(Optional) A RetainSignatureField value that specifies how signature fields are kept in the generated output. Signature fields that are made non-interactive become pictures in the PDF document.

If you provide a literal value, select one of these values:

**None:**

Signed field and unsigned signature fields become non-interactive when the PDF file is flattened.

**All:**

Signed fields and unsigned signature fields remain interactive when the PDF file is flattened.

**Signed Signature Fields:**

Unsigned signature fields become non-interactive. Only Signed signature fields remain interactive when the PDF file is flattened.

**Unsigned Signature Field Only:**

Signed signature fields are made fields become non-interactive. Unsigned signature fields remain interactive when the PDF file is flattened.

**PDFA Specific Render Options properties**

Properties for specifying options when you render a PDF/A document.

*PDF/A Revision Number*

A `string` value that specifies the PDF/A revision number. The only value available is Revision_1.

*PDF/A Conformance*

(Optional) A `PDFAConformance` value that specifies the conformance level with the PDF/A-1 specification to use. Conformance levels indicate how a PDF document adheres to the electronic document preservation requirements for archival and long-term preservation.

If you provide a literal value, select one of these values:

**A:**

(Default) Level A conformance specifies complete conformance with ISO-19005-1:2005. The PDF file is generated by using PDF 1.4 and all colors are converted to either CMYK or RGB. These PDF files can be opened in Acrobat and Adobe Reader 5.0 and later.

**B:**

Level B conformance specifies minimal compliance with ISO-19005-1:2005. The PDF file is generated with all fonts embedded, the appropriate PDF bounding boxes specified, and colors as CMYK or spot colors, or both. Compliant files must contain information describing the printing condition they are prepared for. PDF files created with PDF/X-1a compliance can be opened in Acrobat 4.0 and Adobe Reader 4.0 and later.

**Output properties**

Property to specify the location to store the generated output.

*PDF Output*

The location to store the PDF output. The data type is `document`. For example, select a `document` variable to use to store the PDF document.

**Additional Output properties**

Properties to specify the location to store the results of the operation.

*Meta Data Document*

The location to store the metadata. The data type is `document`.

*Output Result*

The location to store the results of the operation. The data type is `OutputResult`.

**Exceptions**

This operation can throw an OutputException exception.

## generatePrintedOutput operation

Generates a PCL, PostScript, ZPL, IPL, TPCL, or DPL output given a form design and data file. The data file is merged with the form design and formatted for printing. The output generated by this operation can be sent directly to a printer or saved as file. It is recommended that you use this operation when you want to use form designs or data from an application. If your form designs or form designs are located on the network, local file system, or HTTP location, use the generatePrintedOutput operation.

The benefits of using this operation instead of the generatePDFOutput (deprecated) operation are as follows:

- Maximizes application portability because all assets are contained in a single application.
- Simplifies the process design because the form design or data files can be assessed as a `document` variable.

For example, your application requires that you merge a form design with a data file. The data contains hundreds of records. In addition, it requires the output is sent to a printer that supports ZPL. The form

design and your input data are located in an application. Use the generatePrintedOutput operation to merge each record with a form design and send the output to a printer that supports ZPL.

For information about the General and Route Evaluation property groups, see Commonoperation properties.

**Input properties**

Properties to specify a form design, data file, and print format.

*Form*

A `document` value that represents the form design. You create form designs in Designer.

If you provide a literal value, clicking the ellipsis button opens the Select Asset dialog box. (See About-Select Asset.)

*Input Data*

(Optional) A `document` value that specifies the data file that is merged with the form design. The data file that you provide is an XML file.

If you provide a literal value, clicking the ellipsis button opens the Select Asset dialog box. (See AboutSelect Asset.)

*Print Format*

A `PrintFormat` value that specifies the page description language to use, when an XDC file is not provided, to generate the output stream.

If you provide a literal value, select one of these values:

**Custom PCL:**

Use the default XDC file for PCL or specify a custom XDC file for PCL. The default XDC file is hppc5e.xdc.

**Custom PostScript:**

Use the default XDC file for PostScript or specify a custom XDC for PostScript. The default XDC file is ps_plain.xdc.

**Custom ZPL:**

Use the default XDC file for ZPL or specify a custom XDC file for ZPL. The default XDC file is zpl203.xdc.

**Generic Color PCL (5c):**

Use a generic color PCL (5c). The hppcl5c.xdc file is used.

**Generic PostScript Level3:**

Use generic PostScript Level 3. The ps_plain_level3.xdc file is used.

**ZPL 300 DPI:**

Use ZPL 300 DPI. The zpl300.xdc is used.

**ZPL 600 DPI:**

Use ZPL 600 DPI. The zpl600.xdc file is used.

**Custom IPL:**

Use a custom IPL. The default XDC file is ipl203.xdc.

**IPL 300 DPI:**

Use IPL 300 DPI. The ipl300.xdc is used.

**IPL 400 DPI:**

Use IPL 400 DPI. The ipl400.xdc file is used.

**Custom TPCL:**

Use the default XDC file for TPCL or specify a custom XDC file for TPCL. The default XDC file is tpcl203.xdc.

**TPCL 305 DPI:**

Use TPCL 300 DPI. The tpcl305.xdc file is used.

**TPCL 600 DPI:**

Use TPCL 600 DPI. The tpcl600.xdc file is used.

**Custom DPL:**

Use the default XDC file for DPL or specify a custom XDC file DPL. The default XDC file is dpl203.xdc.

**DPL300DPI:**

Use DPL 300 DPI. The dpl300.xdc file is used.

**DPL406DPI:**

Use DPL 400 DPI. The dpl406.xdc is used.

**DPL600DPI:**

Use DPL 600 DPI. The dpl600.xdc is used.

**Template Options properties**

Properties to specify the location to retrieve the XDC file

*Content Root*

(Optional) A `string` value that specifies the URI, absolute reference, or location in the repository to retrieve relative assets used by the form design. For example, if the form design references an image

relatively, such as ../myImage.gif, myImage.gif must be located at repository://. The default value is repository://, which points to the root level of the repository. The default value is `repository://`.

When you pick an asset from your application, the Content Root URI path must have the correct structure. For example, if a form is picked from an application named SampleApp, and is placed at SampleApp/1.0/forms/Test.xdp, the Content Root URI must be specified as`repository://administrator@password/Applications/SampleApp/1.0/forms/`, or`repository:/Applications/SampleApp/1.0/forms/` (when authority is null). When the Content Root URI is specified this way, the paths of all of the referenced assets in the form will be resolved against this URI.

Use the following sources for a URI or absolute reference:

**Repository:**

The repository contains assets that you upload to the AEM Forms Server. The value `repository:///` references the root of the repository. The first two forward slashes are part of the protocol (`repository://`) and the third forward slash represents the root of the repository. For example, if the form design references an image relatively, such as ../myImage.gif, myImage.gif must be located at repository://. The default value is repository://, which points to the root level of the repository.

**Directory in the file system of the AEM Forms Server:**

You can specify a location on the AEM Forms Server, such as `C:\`*[folder name]*. Using a location on the server is not recommended when maximizing the portability of an application.

**Network directory:**

You can specify a location on the network, such as `\\[folder name.]`

**Web location that is accessible by using HTTP:**

After you upload a file to a location on a web server, you can specify the location by using a URL. For example, type `http://`*[server name]*`:`*[port number]*`/`*[folder name]*. The value *[server name]* is the name of the web server, *[port number]* is the port number, and *[folder name]* is the name of the folder.

*XDC URI*

(Optional) A `string` value that specifies the XDC file to use. XDC files are printer description files in XML format. The Output service uses XDC files to output documents to formats such as, PostScript, PCL, DPL, TPCL, and ZPL. You can specify custom XDC files that you create using the XDC Editor. (See AEM Forms XDC EditorHelp

).

If you provide a literal value, you type the name of the XDC file. The property is only editable when you select one of these values in the Print Form property in this operation:

- Custom PCL
- Custom PostScript
- Custom ZPL

- Custom IPL
- Custom TPCL
- Custom DPL

**Output Options properties**

Properties to specify the options to use when generating the output.

*XCI URI*

(Optional) A `string` value that specifies the XCI file to use. XCI files are used to describe fonts that are used for form design elements. XDC files are also used to specify print options such as number of copies, whether duplex printer is used, or stapler options.

*Character Set*

(Optional) A `string` value that specifies the character set used to encode the rendered form.

If you provide a literal value, select the character set to use or select one of these values:

**<Use Server Default>:**

(Default) Select to use the Character Set setting configured on the AEM Forms Server. The Character Set setting is configured using administration console. (See Designer Help.)

**<Use Custom Value>:**

Select and type the canonical name (Java.nio API) of the encoding set that is not in the list. For a complete list of character sets, see http://java.sun.com/j2se/1.5.0/docs/guide/intl/encoding.doc.html.

*Locale*

(Optional) A `string` value that specifies the language used for generating the PDF document.

If you provide a literal value, select a language from the list or select one of these values:

**<Use Server Default>:**

(Default) Use the Locale setting configured on the AEM Forms Server. The Locale setting is configured using administration console. (See Designer Help.)

**<Use Custom Value>:**

After selecting this option, type the Locale ID of the locale code that is not in the list. For a complete list of supported locale codes, see http://java.sun.com/j2se/1.5.0/docs/guide/intl/locale.doc.html.

*Copies*

(Optional) An int value that specifies the number of copies to generate for the output. The default value is 1.

### Batch Options properties

Properties to specify the records name, record levels, and whether to generate multiple outputs for each record in the data file.

*Record Name*

(Optional) A `string` value that specifies element name that identifies the beginning of a batch of records.

*Record Level*

(Optional) An int value that specifies the XML element level that contains the record data.

If you provide a literal value, the default is 1, which represents the first level in the record specified by the Record Name property.

*Generate Multiple Streams*

(Optional) A `boolean` value that specifies whether the operation creates a single or multiple outputs.

If you provide a literal value, the Generate Multiple Streams check box is deselected by default. Select the check box to create multiple outputs. Multiple outputs are useful when you send an output after each record is completed processing. Deselect the check box to create a single output. Single output is useful when you want to send all the processed records at once.

*Enable Lazy Loading*

(Optional) A boolean value that specifies whether incremental (lazy) loading is used when processing multi-record data sets. When incremental loading is used, it helps to reduce the amount of memory used on the AEM forms Server. The use of incremental loading limits XLST options specified in the XCI file because transformations can only be applied to one record at a time.

If you provide a literal value, select the Enable Lazy Loading check box to load and merge one record of data. Deselect the check box to load and merge the entire data file, or all data at one time.

### Rule Options properties

Properties to specify the rules for pattern matching. Pattern matching is used for applying a form design to a specific record that matches a string.

*Pattern Match Size*

(Optional) An int value that specifies the number of bytes to use from the beginning of the input data file to scan for the pattern strings.

*Pattern Matching Rules*

(Optional) A `list` of `string` values for scanning the input data file for a pattern and associates the data with a specific form design. For example, to specify when the pattern matches "cover",to use the form design named CoverLetter.xdp, type `pattern="cover" form="CoverLetter.xdp"`.

If you provide a literal value, click one of these buttons to add or delete an entry from the list.

**Add A List Entry:**

Adds a new rule. After you click this button, a new entry is created in the list. In the Pattern field for the new entry, type a pattern to search for. In the Form field, type the name of the form design for the matching pattern. All the form designs that you specify must be available at the location specified by the Content Root property in this operation.

**Delete A Selected List Entry:**

Removes the selected a rule in the list.

For example, you can specify a pattern such as *car*, and specify the operation to use the form design named *AutoInsurance.xdp*. The form AutoInsurance.xdp is used whenever the text *car* is in the data. For information about working with search rules, Designer Help.

**Destination Options properties**

Properties to specify the location where to send the output.

*Output Location URI*

(Optional) A `string` value that specifies the URI to save the output file. If you create multiple files, the filenames are suffixed with a numeric value. For example, if you specify C:\forms\Loan.pdf, the Output service creates the filenames Loan0001.pdf, Loan0002, Loan0003, and so on. Each of the files are created in the C:\forms folder on the AEM Forms Server.

*Printer Name*

(Optional) A`string` value that specifies the name of the printer for sending the output to for printing. The AEM Forms Server must be configured to access specified printer.

If you provide a literal value, you can specify the name of printer by formatting it as `\\`*[print server]*`\`*[printer name]*. The value *[print server]* represents a printer server and *[printer name]* is the name of the printer.

*LPD URI*

(Optional) A `string` value that specifies the URI of the Line Printer Daemon (LPD) running on the network.

If you provide a literal value, you can specify the name of the LPD by formatting it as `lpd:/`/*[host name]*. The value *[host name]*is the name of the LPD host.

*LPD Printer Name*

(Optional) A`string` value that specifies the name of the printer at the specified Line Printer Daemon (LPD) URI.

If you provide a literal value, you can specify a printer name by formatting it as *[printer name]*. The value *[printer name]* is the name of the printer.

### Printer Options properties

Properties to specify printer options for printing the output.

*Duplex Printing*

(Optional) A `Pagination` value that specifies whether to use two-sided or single-sided printing. Printers that support PostScript and PCL use this value.

If you provide a literal value, select one of these values:

**duplexLongEdge:**

Use two-sided printing and print using long-edge pagination.

**duplexShortEdge:**

Use two-sided printing and print using short-edge pagination.

**simplex:**

Use single-sided printing.

*Staple*

(Optional) A `Staple` value that specifies whether to staple the output on the printer. Printers that support PostScript and PCL use this value.

If you provide a literal value, select of these values:

**Off:**

Do not use the stapler on the printer.

**On:**

Use the stapler on the printer.

**Use Printer Setting:**

Use printer stapler settings.

*Page Offset X*

(Optional) An `string` value that specifies the distance of the page offsets in the horizontal direction. This value is useful when you print the job on preprinted paper and change the origin (for example, the default page offsets for that particular job). This value overrides the page offsets defined in the XDC file. The XDC file settings represent a non-printable area for any PCL device. This option is for PCL devices only.

Use in (inches), cm (centimeter), mm (millimeter) and pt (points) to specify the unit of measure. For example, use type `1 in`.

*Page Offset Y*

(Optional) An `string` value that specifies the distance of the page offsets in the vertical direction. This value is set when you print the job on preprinted paper and change the origin (for example, the default page offsets for that particular job). This value overrides the page offsets defined in the XDC file. The XDC file represents non-printable area for any PCL device. This option is for PCL devices only.

Use in (inches), cm (centimeter), mm (millimeter) and pt (points) to specify the unit of measure. For example, use type `1 in`.

*Output Bin*

(Optional) A `string` value that specifies the output tray on the printer. This value is used to enable the print driver to select the appropriate output bin. For example, type `outputtray1` to specify the first output tray on a printer.

*Output Jog*

(Optional) An `OutputJog` value that specifies that pages that are physically shifted in the output tray.

If you provide a literal value, select one of these values:

**usePrinterSetting**

Use the jog option setting configured on the printer.

**none:**

Do not use jogging.

**pageSet:**

Use jogging each time a set of pages are printed.

**MetaData Options properties**

Properties to specify the format of the metadata files to create.

*Meta Data Spec File*

(Optional) A `string` value that specifies the URI of the metadata spec file to use. A metadata spec file is used to generate metadata from the provided data file.

For example, a metadata spec file can be created as follows:

```
<metadata-spec>
<system>
<map xpath="$fileName" name="fName"/>
<map xpath="$format" name="docType"/>
</system>
<user>
<map xpath="header/txtOrderedByCompanyName" name="companyName"/>
<map xpath="header/txtOrderedByAddress" name="address"/>
<map xpath="header/txtOrderedByCity" name="city"/>
```

```
<map xpath="header/txtOrderedByStateProv" name="state"/>
<map xpath="header/txtOrderedByZipCode" name="zipCode"/>
<map xpath="header/txtOrderedByCountry" name="country"/>
<map xpath="header/txtOrderedByPhone" name="phone"/>
<map xpath="header/txtOrderedByFax" name="fax"/>
</user>
</metadata-spec>
```

If you provide a literal value, clicking the ellipsis button opens the Select Asset dialog box. (See About-Select Asset.)

A metadata file is created with the data that is extracted from each record. The structure of the metadata file that is created is based on the metadata spec file. The resulting metadata file based on the previous metadata spec file is as follows:

```
<root><record id='null'>
<system>
<data name='fName'>c:\cumulativedata0001.pdf</data>
<data name='docType'>PDF</data>
</system>
<user>
<data name='companyName'>Any Company Name</data>
<data name='address'>555, Any Blvd.</data>
<data name='city'>Any City</data>
<data name='state'>Alabama</data>
<data name='zipCode'>12345</data>
<data name='country'>United States</data>
<data name='phone'>(123) 456-7890</data>
<data name='fax'>(123) 456-7899</data>
</user>
</record>
```

*Record ID XPath*

(Optional) An int value that specifies the root level node to use for XPath expressions. The root level node specifies the starting point for XPath expressions. For example, consider a data schema as follows:

```
<batch_100>
<purchaseOrder>
<header>
<txtPONum>1of100</txtPONum>
<dtmDate>2004-02-08</dtmDate>
<txtOrderedByCompanyName>Any Company Name</txtOrderedByCompanyName>
<txtOrderedByAddress>555, Any Blvd.</txtOrderedByAddress>
<txtOrderedByCity>Any City</txtOrderedByCity>
<txtOrderedByStateProv>Alabama</txtOrderedByStateProv>
</header>
</detail
<txtPartNum>580463116</txtPartNum>
<txtDescription>Electric Fuel Pump</txtDescription>
<numQty>1</numQty>
<numUnitPrice>149.95</numUnitPrice>
```

```
<numAmount>149.95</numAmount>
</detail>
</purchaseOrder>
</batch_100>
```

To reference the `<purchaseOrder>` tag as the first level in your XPath expression, set this property to value of 2. To set `<batch_100>` as the first level, set this property to value of 1.

*Generate Record Level Meta Data*

(Optional) A `boolean` value that specifies whether to generate a metadata file for each record.

If a literal value is provided, the Generate Record Level Meta Data check box is deselected. by default. Select the check box to generate a metadata file for each record that is processed. Deselect the check box to generate one metadata file for all records that are processed.

**Output properties**

Property to specify the location to store the generated output.

*Printed Output*

The location to store the PDF output. The data type is `document`.

**Additional Output properties**

Properties to specify the location to store the results of the operation.

*Meta Data Document*

The location to store the metadata. The data type is `document`.

*Output Result*

The location to store the results of the operation. The data type is `OutputResult`.

**Exceptions**

This operation can throw an OutputException exception.

## sendToPrinter operation

Prints a document, such as a PostScript file, using the specified printer protocol.

**TIP:** The Archive wizard can be used to add this operation to a process. (See Preservingprocess results using the Archive Wizard.*)*

For example, your application requires that a record from a data file is merged with a form design, and generates a PostScript file. The PostScript file is then sent to a printer. Use the sendToPrinter operation to send the generated output to a printer.

For information about the General and Route Evaluation property groups, see Commonoperation properties.

### Input properties

Properties to specify the document and destination.

*Input Document*

A `document` value that contains a document output to send to the specified printer. If you are printing a PDF document, the PDF document must be flattened before you to can use this operation.

If you provide a literal value, clicking the ellipsis button opens the Select an Asset dialog box. (See AboutSelect Asset.)

*Printer Protocol*

(Optional) A `PrinterProtocol` value that represents the printer protocol to use.

If you provide a literal value, select one of these values:

**CUPS:**

An indirect method to access the print server. A valid print server must be specified.

**DirectIP:**

A direct method using internet protocols to access the print server that uses the default port 9100. A valid print server must be specified.

**LPD:**

An indirect method to access the specified printer by name through the print server. A valid print server and printer must be specified.

**SharedPrinter:**

A direct method to access the printer directly by name. A valid printer name must be specified.

**NOTE:** *You cannot use the Shared Printer protocol for the sendToPrinter operation when the AEM Forms Server runs on Windows 2008 and has the PDF Generator service deployed. Use alternate protocols like CIFS or Direct IP.*

**CIFS:**

A direct method using the Common Internet File System printing protocol to access a printer or print server. When you use CIFS as a protocol, specify the Universal Naming Convention (UNC) name for the printer or shared printer queue in the Server URI property.

To authenticate with a network printer when using the CIFS protocol, add a password credential to the Trust Store. Use the URL of the printer as the credential name (alias), for example`\\print-server.domain.com\printer`. The credential defines a user name and password that can authenticate against the printer. For information about creating the credential, see Managing certificates and credentials in Administration Help. In a domain with many printers

that share the same credentials, you can also give the the network path of the domain as an alias in the Trust Store. This enables the alias to be used for multiple printers sharing the same credentials in the domain. The Output service looks for the credentials of a printer with URI `\\print-server.domain.name.com\printer` by looking up aliases in the Trust Store in the following order:

```
\\print-server.domain.name.com\printer
\\print-server.domain.name.com
\\domain.name.com
\\name.com
\\com
\\
```

*Server URI*

(Optional) A `string` value that represents the URI of the print server to use that is useful for LPD, CUPS, Direct IP, CIFS, printer protocols. The AEM Forms Server must have access to the print server.

When you use CIFS, use UNC names, such as \\sharedqueue\printername or \sharedqueue.

*Printer Name*

(Optional) A `string` value that represents the name of a particular printer on the specified print server. This value is useful when you set the Printer Protocol property to SharedPrinter or LPD.

**Exceptions**

This operation can throw an OutputException exception.

## transformPDF operation

Converts an interactive PDF form to non-interactive PDF document or PDF/A document. After this conversion, the PDF document can no longer be modified. Use this operation to flatten digitally unsigned PDF documents for printing. When you flatten a PDF document, it is no longer interactive.

For example, your application requires users to fill in a PDF form. After they fill out the form, you want to archive it and make sure it cannot be modified. Use the transformPDF operation to convert the PDF form to a non-interactive form PDF document.

If you provide a flat PDF document as the input for this service, an error is returned that the PDF is already flat. However, if the input document contains both interactive and flat elements, the transformPDF service converts the pages corresponding to the interactive PDF without alerting about the flat PDF.

**NOTE:** When want to convert a PDF document to a PDF/A document, use the Convert toPDF/A operation from the DocConverter service. The toPDFA operation is fully compliant for converting a PDF document containing a digital signature to a PDF/A document.

For information about the General and Route Evaluation property groups, see Commonoperation properties.

### Input properties

Properties to specify the output and PDF conversion options.

*Input PDF Document*

A `document` value that represents a PDF form to convert.

If you provide a literal value, clicking the ellipsis button opens the Select an Asset dialog box. (See AboutSelect Asset.)

*Transformation Format*

A `TransformationFormat` value that specifies the format that the document is rendered to.

If you provide a literal value, select one of these values:

**PDF:**

A non-interactive PDF document is created.

**PDFA:**

A non-interactive PDF/A document is created. PDF/A documents are used for archiving purposes and based on ISO standard 19005-1. It also embeds all the fonts and turns off compression. If you select this value, provide values for the PDF/A Revision Number and PDF/A Conformance properties.

*PDF/A Revision Number*

(Optional) An `int` value that represents the revision number of the specification. This value is always 1.

*PDF/A Amendment*

(Reserved for future) Do not use. A `string` value that represents the amendment number and year separated by a colon.

*PDF/A Conformance*

(Optional) A `PDFAConformance` value that specifies the conformance level with the PDF/A-1 specification to use. Conformance levels indicate how a PDF document adheres to the electronic document preservation requirements for archival and long-term preservation.

If you provide a literal value, select one of these values:

**A:**

(Default) Level A conformance specifies complete conformance with ISO-19005-1:2005. The PDF file is generated by using PDF 1.4 and all colors are converted to either CMYK or RGB. These PDF files can be opened in Acrobat and Adobe Reader 5.0 and later.

**B:**

Level B conformance specifies minimal compliance with ISO-19005-1:2005. The PDF file is generated with all fonts embedded, the appropriate PDF bounding boxes specified, and colors as CMYK or spot colors, or both. Compliant files must contain information describing the printing condition they are prepared for. PDF files created with PDF/X-1a compliance can be opened in Acrobat 4.0 and Adobe Reader 4.0 and later.

### Output properties

Property to represent the output PDF document.

*Output PDF Document*

The location in the process model to save the converted PDF document. The data type is `document`.

### Exceptions

This operation can throw an OutputException exception.

## Output exceptions

The Output service provides the following exceptions for throwing exception events.

### OutputException

Thrown when the service encounters an error during processing of an operation. The following list describes the possible causes for an error:

| Operation | Possible cause |
|---|---|
| generatePDFOutput | The Output service cannot locate the input form. For example, the form name is incorrect, or the repository path that is created from the content root and the form name is incorrect.<br>An incorrect value is provided for the Transformation Format property.<br>The XDP data that is provided for the Input Data property is not valid.<br>The type of data that is provided is not supported.<br>The data that is provided contains malformed XML code.<br>The form that is provided for the Form property is not valid.<br>The file that is provided as the Form value is already a flat PDF document.<br>The Meta Data Spec File defines no user-level meta attributes.<br>The value for Record Name or Record Level does not match the actual name or level in the form data. |

| Operation | Possible cause |
|---|---|
| generatePrintedOutput | The Output service cannot locate the specified form. For example, the form name is incorrect, or the repository path that is created from the content root and the form name is incorrect.<br>The XDP data that is provided for the Input Data property is not valid.<br>The type of data that is provided is not supported.<br>The data that is provided contains malformed XML code.<br>The form that is provided for the Form property is not valid.<br>The file that is provided as the Form value is already a flat PDF document.<br>The Meta Data Spec File defines no user-level meta attributes.<br>The value for Record Name or Record Level does not match the actual name or level in the form data. |
| sendToPrinter | The Output service cannot locate the network printer. For example, the printer name is incorrect, the URI of the print server is incorrect, or the printer is offline.<br>The Output service cannot locate the input document. For example, the path to the document is incorrect.<br>An error occurs on the print device. |
| transformPDF | The Output service cannot locate the specified form or document. For example, the path to the form or document is incorrect.<br>The specified document to transform is not a PDF document.<br>The specified document is not an XFA-based PDF or an Acrobat form.<br>The specified XFA-based PDF form has been reader-extended, digitally signed, or certified. |
| All operations | A required property value is not specified.<br>Services that the Output service depends on are not started (XMLForm, FontManager).<br>The value that is specified for an operation property is not valid. |

## Configuring Output operations to handle all failures

When you add generatePDFOutput and generatePrintedOutput operations to process diagrams, they include exception event throws. However, in some cases, the operations do not throw exception events when they fail. Instead, the returned Status Document includes information about the failure. The Status Document is an XML document that summarizes the results of the operation.

To handle all failure scenarios, include the following configurations in your process diagrams:

- To handle failures that are indicated in the Status Document, draw routes from the operation and add conditions to the route. The conditions evaluate the Status Document to see whether an exception has occurred.
- To handle failures that throw exception events, draw routes from the exception event throws. (See Handlingevent catches.)

The Status Document includes a status element:

- If the operation was successful, the status element contains a zero (0).
- If an exception occurred, the status element contains any text other than a zero (0).

The XPath expression that retrieves the text in the status element is `/process_data/`*[XML variable name]*`/printResult/status`, where *[XML variable name]* is the name of the variable that stores the status document.

Use routing conditions to evaluate the returned status document and determine whether an exception occurred. For example, the Status Document property of a generatePDFOutput operation is set to an XML variable named`XMLvar`. The following condition is true when an exception has occurred:

```
/process_data/XMLvar/printResult/status != 0
```

**Handle exceptions that are indicated in the Status Document:**

1) Configure the generatePDFOutput or generatePrintedOutput operation so that the Status Document property is set to an XML variable. The returned document value will be automatically coerced from a document value (binary) to an XML value (text).
2) Draw a route from the generatePDFOutput or generatePrintedOutput operation to the operation that you want to execute when the exception occurs.
3) Add a condition to the route that compares the text in the status element to zero (0).

**Handle exceptions that throw exception events:**

1) See Handlingevent catches.

# 22.35. PDF Utilities

Interacts with PDF documents so that you can perform tasks such as converting a PDF document to an XDP file, or querying information about a PDF document. For example, you can determine whether a PDF document contains comments or attachments.

For information about using the PDF Utilities service, see Services Reference for AEM Forms

.

## Clone PDF operation

The Clone PDF operation replicates a PDF document. The resulting PDF document can be manipulated independently of the input PDF document. If a given PDF document is passed to multiple services without cloning, the result may be difficult to use effectively.

For example, assume that a PDF document is passed to two services sequentially. When the first service modifies and returns the PDF document as a `document` value, the next service to use the `document` value detects modifications that the first service made.

After using the Clone PDF operation, you are assured that the input `document` value and the result `document` value are identical but distinct, and that any future modification of either value will not be reflected in the other object.

For information about the General and Route Evaluation property groups, see Commonoperation properties.

### Input properties

Property to specify the PDF document.

*PDF*

A `document` value that represents the PDF document to clone.

If you provide a literal value, clicking the ellipsis button opens the Select Asset dialog box. (See About-Select Asset.)

### Output properties

Property to specify the cloned PDF document.

*PDF Clone*

A `document` value that represents the cloned PDF document.

### Exceptions

This operation can throw a PDFUtilityException exception.

## Convert PDF to XDP operation

The Convert PDF to XDP operation converts a PDF document to an XDP file. For a PDF document to be successfully converted to an XDP file, the PDF document must contain an XFA stream in the dictionary.

For information about the General and Route Evaluation property groups, see Commonoperation properties.

### Input properties

Property to specify the PDF document to convert.

*Input Document*

A `document` value that represents the PDF document to convert to an XDP file.

If you provide a literal value, clicking the ellipsis button opens the Select Asset dialog box. (See AboutSelect Asset.)

**Output properties**

Property to specify the converted XDP file.

*Converted XDP*

A `document` value that represents an XDP file.

**Exceptions**

This operation can throw a PDFUtilityException exception.

## Convert XDP to PDF operation

The Convert XDP to PDF operation converts an XDP file to a PDF file. To successfully convert an XDP file to a PDF file, the XDP file must contain an encoded PDF packet.

For information about the General and Route Evaluation property groups, see Commonoperation properties.

**Input properties**

Property to specify the XDP file to convert.

*InDoc*

A `document` value that represents the XDP file to convert to a PDF file.

If you provide a literal value, clicking the ellipsis button opens the Select Asset dialog box. (See AboutSelect Asset.)

**Output properties**

Property to specify the converted PDF document.

*Converted PDF*

A `document` value that represents the PDF document that was converted.

**Exceptions**

This operation can throw a PDFUtilityException exception.

## Get PDF Properties operation

The Get PDF Properties operation performs queries on the specified PDF document and returns the results as a `PDFPropertiesResult` value.

For information about the General and Route Evaluation property groups, see Commonoperation properties.

### Input properties

Property to specify the PDF document to query.

*PDF*

A `document` value that represents the PDF document to query.

*Query Options*

(Optional) A `PDFPropertiesOptionSpec` value that specifies which queries are performed. If you provide a literal value, set one or more of the following `boolean` properties to form the query:

- Is a PDF Document
- Is a PDF Package
- Get the PDF Version
- Check for Attachments
- Check for Comments
- Recommended Acrobat Version
- Form Type
- Check for AcroForm
- Has a Fillable Form
- Is an XFA Document
- Get the XFA Version

  *A value of `null` means that either the property was not requested or could not be determined.*

### Output properties

Property to specify the result of the query.

*PDF Properties*

A `PDFPropertiesResult` value that contains the result of the query.

### Exceptions

This operation can throw a PDFUtilityException exception.

## Get PDF Save Mode operation

The Get PDF Save Mode operation returns the save mode of a PDF document. The save mode represents the mode in which the PDF document is saved. In addition, the save mode specifies whether the request is considered a requirement or only a suggestion. Save mode values are not influenced by the PDF document content.

For information about the General and Route Evaluation property groups, see Commonoperation properties.

### Input properties

Property to specify the PDF document.

*PDF*

A `document` value that represents the PDF document for which save mode information is returned.

If you provide a literal value, clicking the ellipsis button opens the Select Asset dialog box. (See AboutSelect Asset.)

### Output properties

Property to specify the save mode.

*Save Mode*

A `PDFUtilitySaveMode` value that represents the save mode of the PDF document. The following values are possible PDF save-mode values:

- `FAST_WEB_VIEW`, which is used while viewing the PDF document online.
- `INCREMENTAL`, which performs the save operation in the least amount of time.
- `FULL`, which saves with fewer optimizations.

### Exceptions

None.

## Multiple Clones of PDF operation

The Multiple Clones of PDF operation clones a PDF document a specified number of times. The resulting PDF documents are used independently of the input PDF document.

For information about the General and Route Evaluation property groups, see Commonoperation properties.

### Input properties

Properties to specify the PDF document to clone and the number of clones.

*PDF*

A `document` value that represents the PDF document to be cloned.

If you provide a literal value, clicking the ellipsis button opens the Select Asset dialog box. (See AboutSelect Asset.)

*Number Of Copies*

An `int` value of the number of clones to create.

**Output properties**

Property to specify the cloned PDF documents.

*Array of PDF Clones*

A `list` of `document` objects to store one or more cloned PDF documents.

**Exceptions**

This operation can throw a PDFUtilityException exception.

## Set PDF Save Mode operation

The Set PDF Save Mode sets the save mode of a PDF document. The save mode represents the mode in which the PDF document is saved. In addition, the save mode specifies whether the request is considered a requirement or only a suggestion. Save mode values are not influenced by the PDF document content.

For information about the General and Route Evaluation property groups, see Commonoperation properties.

**Input properties**

Properties to specify the PDF document and the save mode.

*PDF*

A `document` value that represents the PDF document for which save mode information is to be set.

*Save Mode*

A `PDFUtilitySaveMode` object that specifies the save mode of the PDF document. If you provide a literal value, set the following options.

**Save Required:**

Check this option if saving is required.

**Save Style:**

The following values are valid PDF save mode values:

- `FULL`, which saves with fewer optimizations.
- `INCREMENTAL`, which performs the save operation in the least amount of time.
- `FAST WEB VIEW`, which is used while viewing the PDF document online.

*Override*

(Optional) A `boolean` value that specifies whether to make the setting regardless of any previous requests. If `True`, the save mode is always set. If `False`, the save mode is set only if a save mode was previously set on the document.

**Output properties**

Property to specify the PDF document.

*PDF*

A `document` value that represents the PDF document for which the save mode information has been set.

**Exceptions**

None.

## Redact PDF operation

Redacting a document means removing portions from the document which could contain sensitive information such as text, pictures, or watermarks. Acrobat includes a post-redaction option for removig hidden information such as hidden text and metadata. However, with the redaction operation in AEM Forms, you can remove only displayable contents.

Before you redact displayable portions of a PDF document, you can use the redaction tools in Acrobat to annotate the regions of the document that must be redacted and optionally, specify any post-redaction mark.

**Input properties**

Properties to specify the parameters of the redaction operation.

*PDF*

A `document` value that specifies the PDF document to be redacted. The PDF document must be already annotated for the redaction operation to be successful.

*Redaction Options*

A `RedactionOptionSpec` object that specifies the redaction options. If you specify a literal value, specify the following:

**Redact XObject references**

Select this option if you want to redact all references of the same XObject.

**Redact whole image for unsupported filter**

Select this option if you want to redact whole image when only parts of the image are marked for redaction but the image format is not supported by the image filters.

**Output properties**

A `RedactionResult` object that specifies the output of the redaction operation, containing the following:

*PDF*

A `document`value that specifies the document resulting after the redaction operation.

*Redaction Status*

A `boolean` value that specifies whether the redaction operation was successful. A true value indicates that the PDF document was redacted successfully. If the redaction operation was unsuccessful or partially successful, a PDFUtilityException is thrown.

### PDF Utilities exceptions

The PDF Utilities service provides the following exceptions for throwing exception events.

**PDFUtilityException**

Thrown if an error occurs during a PDF Utilities service operation.

## 22.36. Process Engine Connector for IBM FileNet

Lets you create processes that interact with IBM FileNet Workflow Step stored in the FileNet Process Engine. (See IBM FileNet P8 documentation

.)

### Dispatch Workflow Step operation

Takes a FileNet Workflow Step from the specified queue and dispatches it. This operation is used to advance a workflow object in the FileNet Workflow.

For information about the General and Route Evaluation property groups, see Commonoperation properties.

### Login Settings properties

Login settings for invoking the FileNet Process Engine Connector service.

*User Name*

A `string` value that specifies the account name to use when connecting to the FileNet Process Engine.

*Password*

A `string` value that specifies the password that is associated with the account name specified in the User Name property to use when connecting to the FileNet process engine.

### FileNet Workflow Step Information properties

*Queue Name*

A `string` value that represents the name of the queue where the FileNet Workflow Step is located.

*Work Object Number*

A `string` value that represents the Work Object Number of a FileNet Workflow Step.

### Exceptions

This operation can throw a RepositoryException exception.

## Retrieve Workflow Step Parameters operation

Retrieves the FileNet Workflow Step parameters from a process running on AEM Forms.

For information about the General and Route Evaluation property groups, see Commonoperation properties.

### Login Settings properties

Login settings for invoking the FileNet Process Engine Connector service.

*User Name*

A `string` value that specifies the account name to use when connecting to the FileNet Process Engine.

*Password*

A `string` value that specifies the password that is associated with the account name specified in the User Name property to use when connecting to the FileNet process engine.

### FileNet Workflow Step Information properties

Properties to specify the IBM FileNet Workflow Step to retrieve.

*Queue Name*

A `string` value that represents the name of the queue where the FileNet Workflow Step is stored.

*Work Object Number*

A `string` value that represents Work Object Number of the FileNet Workflow Step.

**FileNet Workflow Step Parameters properties**

Properties to specify the IBM FileNet Workflow Step to retrieve.

*Object Store*

(Optional) A `string` value that represents the name of the FileNet repository. You can choose from the list of available repositories from your current connection broker defined in administration console. (See FileNet administration help

.)

*Workflow Definition GUID Or Path*

(Optional) A `string` value that represents the GUID or path of the Workflow Definition file in the IBM FileNet repository.

If you provide a literal value, type the path or click Browse to select the folder from the FileNet repository.

When typing the folder path, use the following rules:

- Do not include the object store name in the folder path.
- Because the use of attachments is not supported, the path cannot reference an attachment.
- Use a forward slash (/) to separate folder names if you specify a folder location, as shown in this example:

  ```
  /[FolderPath]/[Workflow Definition]
  ```

*Workflow Map Name*

(Optional) A `string` value that species the Workflow Map that the FileNet Workflow Step is part of.

*Workflow Step ID*

(Optional) A `string` value that represents identifier of the FileNet Workflow Step.

*Step Parameter Mapping Table*

A `map` value that maps a list of FileNet Workflow Step parameters to process variables. During parameter retrieval, the mapped process variables are set with the values from the corresponding FileNet Workflow Step parameters.

If you provide a literal value, follow these steps to populate the table:

1) Specify GUID or path of the Workflow Definition file in the Workflow Definition GUID or Path box.
2) Select a step ID from the Workflow Step ID list.
3) Repeat step 1 and 2 if required.

**Result properties**

*Step Parameter Name Value Map*

The location to store the Workflow Step parameters from the IBM FileNet process engine. The data type is `map`.

**Exceptions**

This operation can throw a RepositoryException exception.

## Set Workflow Step Parameters operation

Sets the parameters in a FileNet Workflow Step from within a process running on AEM Forms.

For information about the General and Route Evaluation property groups, see Commonoperation properties.

**Login Settings properties**

Login settings for invoking the FileNet Process Engine Connector service.

*User Name*

A `string` value that specifies the account name to use when connecting to the FileNet Process Engine.

*Password*

A `string` value that specifies the password that is associated with the account name specified in the User Name property to use when connecting to the FileNet process engine.

**FileNet Workflow Step Information properties**

Properties to specify the FileNet Workflow Step to set.

*Queue Name*

A `string` value that represents the name of the queue where the FileNet Workflow Step is stored.

*Work Object Number*

A `string` value that represents identifier of the FileNet Workflow Step.

### FileNet Workflow Step Parameters properties

Parameters to set within the FileNet Workflow Step.

*Object Store*

(Optional) A `string` value that represents the name of the FileNet repository. You can choose from the list of available repositories from your current connection broker defined in administration console.

*Workflow Definition GUID Or Path*

(Optional) A `string` value that represents the GUID or path of the Workflow Definition file in the IBM FileNet repository.

If you provide a literal value, type the path or click Browse to select the folder from the FileNet repository.

When typing the folder path, use the following rules:

- Do not include the object store name in the folder path.
- Because the use of attachments is not supported, the path cannot reference an attachment.
- Use a forward slash (/) to separate folder names if you specify a folder location, as shown in this example:

  ```
  /[FolderPath]/[Workflow Definition]
  ```

*Workflow Map Name*

(Optional) A `string` value that species the Workflow Map that the FileNet Workflow Step is part of.

*Workflow Step ID*

(Optional) A `string` value that represents identifier of the FileNet Workflow Step.

*Step Parameter Mapping Table*

A `map` value that maps a list of IBM FileNet Workflow Step parameters to process variables. During parameter setting, the mapped Workflow Step parameters are set with the values from the corresponding process variables.

If you provide a literal value, follow these steps to populate the table:

1) Specify GUID or path of the Workflow Definition file in the Workflow Definition GUID or Path box.
2) Select a step ID from the Workflow Step ID list.
3) Repeat step 1 and 2 if required.

### Exceptions

This operation can throw a RepositoryException exception.

## Process Engine Connector for IBM FileNet exceptions

The Process Engine Connector for IBM FileNet service provides the following exceptions for throwing exception events.

### RepositoryException

Thrown if an error occurs while retrieving or storing a document in a FileNet repository (for example, if the value of an operation parameter is `NULL` or if FileNet returns an error).

# 22.37. Print PDF Package

Print PDF Package is the service for the PrintPDFPackage process that AEM Forms Output 9 provides.

The process uses the Assembler, Output, and PDF Utilities services to convert a PDF package to a PostScript file that is suitable for printing.

A PDF package contains multiple documents wrapped in one PDF file. For more information on PDF packages, see the documentation for Acrobat 8.

## Invoke operation

Converts a PDF package to a PostScript file.

For information about the General and Route Evaluation property groups, see Commonoperation properties.

### Input properties

Property for specifying the input document.

*InPDFDoc*

A `document` value that represents a PDF package, a flat PDF file, or a PDF form.

### Output properties

Property for specifying the output document.

*OutPSDoc*

The location to store the PostScript document that is suitable for printing. The data type is `document`.

### Invocation Policy properties

Specifies whether to wait for a response after invoking the operation by selecting either Do Not Wait For Response or Wait For Response. The default is `Wait For Response`.

*Do Not Wait For Response*

Specifies that no response is required from the server.

*Wait For Response*

Specifies that a response is required from the server before further processing.

# 22.38. Acrobat Reader DC extensions

Acrobat Reader DC extensions allows you to manipulate usage rights on PDF documents. Usage rights pertain to functionality that is available in Acrobat but not in Adobe Reader. Functionality controlled by Acrobat Reader DC extensions includes the ability to add comments to a document, fill forms, and save the document. PDF documents that have usage rights added are called *rights-enabled documents*. A user who opens a rights-enabled PDF document in Adobe Reader can perform the operations that are enabled for that document.

In addition to PDF documents, you can use Acrobat Reader DC extensions operations to manipulate usage rights for PDF forms.

**IMPORTANT:** When you use Acrobat Reader DC extensions to add usage rights to a static PDF form for use with Workspace, save the form by clicking Take Offline. This step preserves the usage rights. Clicking the Save button in Adobe Reader causes the usage rights to be unavailable to the user.

For information about using the Acrobat Reader DC extensions service, see Services Reference for AEM Forms

.

## Apply Usage Rights operation

Applies usage rights to a PDF document. Before you use the Apply Usage Rights operation, import and configure a Acrobat Reader DC extensions Credential. (See Configuring credentials for use with Acrobat Reader DC extensions in*AEM Forms administration help*

.)

If you are applying usage rights to a PDF form for use in Workspace, inject the Form Bridge library as well. Perform the steps in the following order:

1) Create the PDF using, for example, renderPDFForm (Forms service).
2) Use InjectForm Bridge (Form Augmenter service) to add the Form Bridge library.
3) Apply usage rights.

   *If you inject the Form Bridge libraries after applying usage rights, the usage rights are invalidated. Any combination of encrypting, certifying, and applying usage rights to the same document must occur in the following order. These services must be invoked within a short-lived process:*
4) Encrypt (Encryption service or ProtectDocument operation of the Rights Management service)
5) Certify (Signature service)

6) Apply usage rights

For information about the General and Route Evaluation property groups, see Commonoperation properties.

**Input properties**

Properties to specify the PDF document, the credential alias, and usage rights to apply to the PDF document.

*Input PDF Document*

A `document` value that represents the PDF document for which to apply usage rights.

If you provide a literal value, clicking the ellipsis button opens the Select Asset dialog box. (See About-Select Asset.)

*Credential Alias*

A `string` value that represents the alias of the credential to grant usage rights.

If you provide a literal value, from the Credential Alias list, select the identifier of a credential in the trust store.

*Apply Usage Rights Options*

(Optional) A `ReaderExtensionsOptionSpec` value that represents the usage rights to apply to the PDF document.

If you provide a literal value, you can apply any of the following usage rights to the PDF document.

**Basic Form Fill-in:**

Select to permit basic form fill-in capabilities in the PDF document.

**Import and Export Form Data:**

Select to permit users to import or export form data.

**Submit Outside Web Browser:**

Select to permit users to submit form data by email or offline.

**Database and Web Service Connectivity:**

Select to permit a user to access to the database or call the web service that is defined within the PDF document.

**Add, Delete, And Change Form Fields:**

Select to permit users to edit existing filled form fields in the PDF document.

**Create Pages From Templates:**

Select to permit users to add new pages generated by existing templates in the PDF document.

**2D Barcode Decoding:**

Select to permit two-dimensional (2D) barcode decoding in the PDF document.

**Digital Signatures:**

Select to permit users to add digital signatures to the PDF document.

**Commenting:**

Select to permit offline commenting of the PDF document.

**Online Commenting:**

Select to permit online commenting of the PDF document.

**Embedded File Attachments:**

Select to permit users to add attachments to the PDF document.

**Count Credential Usage:**

Select Yes to create a count of how many times the credential is used to enable usage rights. The count appears in the Acrobat Reader DC extensions web application.

**Reader Message:**

A message you type that represents the text displayed within Adobe Reader to inform users that the PDF document contains usage rights.

### Output properties

Property to specify the PDF document with usage rights applied.

*Output PDF Document*

(Optional) The location to store the PDF document with usage rights applied. The data type is `document`.

### Exceptions

This operation can throw a ReaderExtensionsException exception.

## Get Document Usage Rights operation

Retrieves information about usage rights that are applied to a PDF document.

For information about the General and Route Evaluation property groups, see Commonoperation properties.

### Input properties

Property to specify the PDF document for which to retrieve usage rights information.

*Input PDF*

A `document` value that specifies the PDF document for which to retrieve usage rights information.

**Output properties**

Property to specify the destination for the usage rights information.

*Document Usage Rights*

(Optional) The location to store usage rights information for the specified PDF document. The data type is `GetUsageRightsResult`.

**Exceptions**

This operation can throw a ReaderExtensionsException exception.

## Get Credential Usage Rights operation

Retrieves information about the specified credential, including the usage rights it extends to a document.

For information about the General and Route Evaluation property groups, see Commonoperation properties.

**Input properties**

Property to specify the credential for which to retrieve usage rights information.

*Credential Alias*

A `string` value that specifies the credential for which to retrieve usage rights information.

If you provide a literal value, from the Credential Alias list, select the identifier of a credential in the trust store.

**Output properties**

Property to specify the destination for the usage rights information.

*Credential Usage Rights*

(Optional) The location to store the usage rights information for the specified credential. The data type is `GetUsageRightsResult`.

**Exceptions**

This operation can throw a ReaderExtensionsException exception.

## Remove Usage Rights operation

Removes usage rights from the specified PDF document.

For information about the General and Route Evaluation property groups, see Commonoperation properties.

**Input properties**

Property to specify the rights-enabled PDF document.

*Input PDF Document*

A `document` value that specifies the PDF document from which to remove usage rights.

If you provide a literal value, clicking the ellipsis button opens the Select Asset dialog box. (See AboutSelect Asset.)

**Output properties**

Property to specify the PDF document without usage rights.

*Output PDF Document*

(Optional) The location to store the PDF document without usage rights. The data type is `document`.

**Exceptions**

This operation can throw a ReaderExtensionsException exception.

### Acrobat Reader DC extensions exceptions

The Acrobat Reader DC extensions service provides the following exception for throwing exception events.

**ReaderExtensionsException**

Thrown when an error occurs while adding usage rights to or removing usage rights from a PDF document.

## 22.39. Render Form Guide

Renders a form, created in Designer 8.2, to a form guide, provided that a form guide layout was created using Guide Builder. You can use this service in a process map or to configure the render service, which is a preprocessing step for an xfaForm variable.

It is not recommended that you modify this service directly because it is possible to do so by using Workbench. Make a copy of the service, modify it, and then use your copy as a preprocessing step for an xfaForm variable.

**IMPORTANT:** Upgrading to AEM Forms leaves the Render Form Guide service unchanged. If you want to change your previous render service, then you need to import that service.

## invoke operation

Sets the necessary parameters for retrieving the form, specifies the location of the form guide, and invokes the renderFormGuide operation.

For information about the General and Route Evaluation property groups, see Commonoperation properties.

### Input properties

*Form Url*

A `string` value that specifies the URL location of the form in the repository or at a network location on a file system.

*Input Form Data*

A `document` value that represents the form data to merge with the form.

If you provide a literal value, when you click the ellipsis button , the Open dialog box appears, where the form that represents the data can be selected from the local computer or a network location.

*Target Url*

A `string` value that specifies the URL of the web application or servlet that the submitted form data is sent to.

*User Agent*

A `string` value that specifies a list of the user's web browser settings, delimited by a semicolon (;). For example, such a list may appear like the following value:

```
Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1; .NET CLR
2.0.50727;
```

### Output properties

*Rendered Form*

A `document` value that stores the location of the form guide.

### Invocation Policy properties

Specifies whether to wait for a response after invoking the operation by selecting either Do Not Wait For Response or Wait For Response. The default is `Wait For Response`.

*Do Not Wait For Response*

Specifies that no response is required from the server.

*Wait For Response*

Specifies that a response is required from the server before further processing.

# 22.40. Render HTML Form

Renders a form created in Designer to an HTML form that is usable within Workspace. You can use this service in a process map or to configure the render service of an xfaForm variable.

It is not recommended that you modify this service directly as it is possible to do so with Workbench. It is recommended that you make a copy of the service, modify it, and then use your copy as a preprocessing step for an xfaForm variable.

**IMPORTANT:** Upgrading to AEM Forms leaves the Render HTML Form service unchanged. If you want to change your previous render service, then you need to import that service.

**NOTE:** In Designer, special form objects must be added to the form before it can be rendered as an HTML form for use with Workspace. For information about how to add a form guide and how to prepare a form for Process Management, see Using Designer > Creating Forms for AEM Forms Process Management in the Designer Help.

**NOTE:** For a form design, it is recommended that the guidelines for layout considerations for HTML forms be followed to overcome text shift issues in a rendered HTML form. (See Using Designer > Creating Forms for Forms > Creating HTML forms > Layout considerations for HTML forms in the Designer Help.)

## invoke operation

Sets the necessary parameters for retrieving the form, specifies the location to access the HTML form, enables the form for offline usage within Workspace, and invokes the renderHTMLForm(deprecated) operation.

For information about the General and Route Evaluation property groups, see Commonoperation properties.

### Input properties

*Form Url*

A `string` value that specifies the URL location of the form in the repository or at a network location on a file system.

*Input Data*

A `document` value that represents the form data to merge with the form.

If you provide a literal value, when you click the ellipsis button , the Open dialog box appears where the file that represents the data can be selected from the local computer or a network location.

*Server Email*

A `string` value that specifies the email address to respond to.

*Target Url*

A `string` value that specifies the URL of the web application or servlet that the submitted HTML form is sent to.

*Task Id*

A `string` value that specifies the unique identifier for the task.

*User Agent*

A `string` value that specifies a list of the user's web browser settings, delimited by a semicolon (;). For example, such a list may appear like the following value:

```
Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1; .NET CLR
2.0.50727;
```

**Output properties**

*Output Form*

A `document` value that stores the location of the HTML form that this operation creates.

**Invocation Policy properties**

Specifies whether to wait for a response after invoking the operation by selecting either Do Not Wait For Response or Wait For Response. The default is `Wait For Response`.

*Do Not Wait For Response*

Specifies that no response is required from the server.

*Wait For Response*

Specifies that a response is required from the server before further processing.

# 22.41. Render PDF Form

Renders a form created in Designer to a PDF form that is usable within Workspace. You can use this service in a process map or to configure the render service, which is a preprocessing step for an xfaForm variable.

It is not recommended that you modify this service directly because it is possible to do so by using Workbench. It is recommended that you make a copy of the service, modify it, and then use your copy as a preprocessing step for an xfaForm variable. For example, you may want to modify this service to use certified signatures with Workspace.

**IMPORTANT:** Upgrading to AEM Forms leaves the Render PDF Form service unchanged. If you want to change your previous render service, then you need to import that service.

## invoke operation

Sets the necessary parameters for retrieving the form, specifies the location of the PDF form, enables the form for offline and online use with Workspace, and invokes the renderPDFForm operation.

For information about the General and Route Evaluation property groups, see Commonoperation properties.

### Input properties

*Form Url*

A `string` value that specifies the URL location of the form in the repository or at a network location on a file system.

*Input Data*

A `document` value that represents the form data to merge with the form.

If you provide a literal value, when you click the ellipsis button , the Open dialog box appears, where the file that represents the data can be selected from the local computer or a network location.

*Server Email Address*

A `string` value that specifies the email address to respond to. This property is used only when the user is submitting the form offline.

*Target Url*

A `string` value that specifies the URL of the web application or servlet that the submitted form data is sent to.

*Task Id*

A `string` value that specifies the unique identifier for the task.

*User Agent*

A `string` value that specifies a list of the user's web browser settings, delimited by a semicolon (;). For example, such a list may appear like the following value:

```
Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1; .NET CLR
2.0.50727;
```

### Output properties

*Rendered Form*

The location to store the PDF form that is rendered. The data type is `document`.

### Invocation Policy properties

Specifies whether to wait for a response after invoking the operation by selecting either Do Not Wait For Response or Wait For Response. The default is `Wait For Response`.

*Do Not Wait For Response*

Specifies that no response is required from the server.

*Wait For Response*

Specifies that a response is required from the server before further processing.

# 22.42. Repository

Reads content, stored as resources, from the repository. The repository stores resources such as forms and their related assets (schemas, fragments, images, and so on) that are used within services and provides automatic versioning of all resources. The resources in the repository are managed in the Resources view. (See ManagingResources.)

For information about using the Repository service, see Services Reference for AEM Forms

.

## Read Resource Content operation

Specifies the content to read from the repository.

For information about the General and Route Evaluation property groups, see Commonoperation properties.

### Input properties

Property for specifying the file path of the content.

*Resource Uri*

A `string` value that specifies the path to the form located on the repository.

To provide a literal value, you specify the path to the resource. For example, for a file named document.pdf that is located in a top-level folder named documents, the path to the file is `/documents/document.pdf`.

You can either type the path, or click Browse to select the resource.

### Output properties

Property for specifying the location to store the content read from the repository.

*Result*

The location to store the retrieved item. The data type is `document`.

**Exceptions**

This operation can throw a RepositoryException exception.

### Repository Exceptions

**RepositoryException**

Thrown when an error occurs while retrieving the item from the repository.

## 22.43. Document security

Applies, removes, and modifies policy protection on PDF documents. You can secure documents by creating policies, editing existing policies, removing security from documents, and obtaining the licenses for policy-protected documents.

For information about using the document security service, see Services Reference for AEM Forms

.

### Apply policy operation (deprecated)

Secures the specified PDF document with an existing policy. This operation is deprecated in LiveCycle ES2. Use ProtectDocument instead.

For information about the General and Route Evaluation property groups, see Commonoperation properties.

**Input properties**

Properties to specify the PDF document to which to apply the policy, the document name, and the policy set. Optionally, you can specify the user profile and domain of the publisher.

*Input PDF Document*

A `document` value that represents the PDF document to secure with the policy.

If you provide a literal value, clicking the ellipsis button opens the Select Asset dialog box. (See About Select Asset.)

*Document Name*

A `string` value that contains the name of the document. This value can have a maximum length of 50 single-byte characters and represents the name of the document audited by the document security service.

If you provide a literal value, type the name of the document.

*Policy Set Name*

(Optional) A `string` value that specifies the policy set to use for securing the document. The value of `My Policies` is used if a policy set name is not provided.

If you provide a literal value, type the name of the policy set.

*Policy Name*

A `string` value that specifies the name of the policy to use to secure the document.

If you provide a literal value, type the name of the policy.

*Publisher Username*

(Optional) A `string` value that represents the user name of the User Manager user who is the publisher of the document. The Document Publisher controls the policy associated with a document. The Document Publisher's identity is used to apply a policy to a document and has special permissions to remove a policy from a document or switch a policy on a document.

To use this property, specify a value for the Publisher Domain property as well.

For long-lived services, provide both Publisher Username and Publisher Domain values.

For short-lived services, if you do not provide values, the user name and domain are picked up from the user account that is used to invoke the process.

If you provide a literal value, type the publisher's user profile.

*Publisher Domain*

(Optional) A `string` value that represents the name of the User Manager domain of the user who is the publisher of the document. To use this property, specify a value for the Publisher Username property as well.

If you provide a literal value, type the name of the publisher's domain.

**Output properties**

Property for specifying the PDF document that the policy is applied to.

*Result PDF Document*

(Optional) The policy-protected PDF document. The data type is `document`.

### Exceptions

This operation can throw an SDKException exception.

## Create policy from existing policy operation

Creates a policy based on an existing policy. This operation can also update the validity period, principals, permissions, and watermark.

The new policy and the existing policy must be from the same type of policy set. For example, to create a new policy in a My Policy policy set, the existing policy must also be from a My Policy policy set.

For information about the General and Route Evaluation property groups, see Commonoperation properties.

### Policy properties

Properties that specify the source, policy set, and name of the new policy.

*(From) Policy Set*

(Optional) A `string` value that specifies the name of a policy set that contains the policy to use to create a policy. The value of `My Policies` is used if a policy set name is not provided.

If you provide a literal value, type the name of the policy set.

*Policy Name*

A `string` value that specifies the name of the policy to use to create a policy.

If you provide a literal value, type the name of the source policy.

*(To) Policy Set to Include New Policy*

(Optional) A `string` value that specifies the policy set for the new policy. The value of `My Policies` is used if a policy set name is not provided.

If you provide a literal value, type the name of the policy set.

**NOTE:** This policy set must be of the same type as the one specified for the (From) Policy Set property. For example, if the (From) Policy Set policy is from a My Policies policy set, then this set must also be a My Policies policy set.

*New Policy Name*

A `string` value that specifies the name of the new policy. This value can have a maximum length of 50 characters.

If you provide a literal value, type the name of the policy.

### Validity properties

Properties that specify the length of time a user can access a document offline and the time during which a document is valid.

*Offline Lease Period*

(Optional) A `string` value that represents the number of days a recipient can use the document without a network connection to the document security server. The default is `0`.

If you provide a literal value, type the number of days.

*Policy Validity Starting Date*

(Optional) A `date` value that represents when the policy becomes valid.

If you provide a literal value, specify No Starting Date or select a date and time by using the calendar button and the Starting Time box.

*Policy Validity Expiration Date*

(Optional) A `date` value that represents when the policy becomes invalid.

If you provide a literal value, specify No Expiration Date or select a date and time by using the calendar button and the Expiration Time box.

### Principals properties

Properties that specify principals to add to or remove from the new policy.

*Principals To Add*

(Optional) A `Userlist` value that represents the principals to add to the new policy. If a value is not provided, only the principals from the source policy are used. Each principal is given the permissions specified by the Permissions properties.

If you provide a literal value, click the plus sign button to add a principal. You can specify principals as a user list, a user or group selected by name or email address, a variable, or an XPath expression. Click the minus sign button to remove a principal from the list. (See AboutSelect User and AboutSelect Group.)

*Principals To Remove*

(Optional) A `Userlist` value that represents the principals to exclude from the new policy. The principals that you specify are removed from the list of principals in the source policy.

If you provide a literal value, click the plus sign button to remove a principal. You can specify principals as a user list, a user or group selected by name or email address, a variable, or an XPath expression. Click the minus sign button to remove a principal from the list. (See AboutSelect User and AboutSelect Group.)

### Permissions properties

Properties that specify the actions that are available to users when the new policy is applied to a document. Specify the users in the Principals properties.

*Permissions for the Policy*

(Optional) A `list` value that adds permissions to the new policy.

If you provide a literal value, you can configure the following permissions:

- In the Print list, select a permitted action. The default is Allowed.
- In the Change list, select the permitted action. The default is Any.
- Select Offline to allow users to view the document while disconnected from the internet. Default is selected.
- Select Copy to allow users to make a copy of the document. Default is selected.
- Select Screen Reader to enable screen reader devices for visually impaired users to access the text in documents. Default is selected. Not available if the Copy option is selected.

### Watermark properties

Properties that specify a watermark for the new policy.

*Watermark Name*

(Optional) A `string` value that represents the name of the watermark used by the new policy.

If you provide a literal value, in the Watermark list, select No Watermark or a watermark value. Configure watermarks in document security in the Administration Console. (See Document security administration help

.)

### Exceptions

This operation can throw an SDKException exception.

## Create policy from template operation (deprecated)

Creates a policy based on the specified policy template. This operation is deprecated in LiveCycle ES2. Use Createpolicy from existing policy instead.

For information about the General and Route Evaluation property groups, see Commonoperation properties.

### Input properties

Properties to specify the PDF document for which to apply the policy, the document name, and the policy to apply to the PDF document. Optionally, you can specify the user profile and group profile of the publisher.

*New Policy Set Name*

(Optional) A `string` value that represents the policy set used to secure the policy with. The value of `My Policies` is used if a policy set name is not provided.

If you provide a literal value, type a new name for the policy set.

*New Policy Name*

A `string` value that contains the new name of the policy. This value can have a maximum length of 50 characters.

If you provide a literal value, type the name of the document.

*Policy Set Name*

(Optional) A `string` value that specifies the name of the policy set to be used as a template for securing the PDF document. The value of `My Policies` is used if a policy set name is not provided.

If you provide a literal value, type the name of the policy set.

*Policy Name*

A `string` value that specifies the name of the policy template to use to create a policy.

If you provide a literal value, type the name of the policy template.

*Offline Lease Period*

(Optional) A `string` value that represents the number of days a recipient can take the document offline and use it without a network connection. The default is `30`.

If you provide a literal value, type the number of days.

*Principals To Add*

(Optional) A `list` of `Principal Reference` values that represent the principals to add. Each principal is given the following permissions:

**Open online:**

Users can view the document while connected to the internet.

**Open offline:**

Users can view the document while disconnected from the internet.

**Accessible:**

Enables screen reader devices for visually impaired users to access the text in documents.

You cannot configure the actions or permissions of the principals. If a value is not provided, only the principals from the template are used.

If you provide a literal value, click the plus sign button to display the Add Users and Groups dialog box to add principals. You can also select a principal from the list and click the minus sign button to remove it.

*Principals To Remove*

(Optional) A `list` of `Principal Reference` values (`com.adobe.idp.um.api.infomodel.PrincipalReference`) that represent the principals to remove. The principals that you specify are removed from the set of principals that are in the template policy that you are copying from.

If you provide a literal value, click the plus sign button to display the Add Users and Groups dialog box to add principals. You can also select a principal from the list and click the minus sign button to remove it. (See AboutAdd Users and Groups.)

*Watermark Name*

(Optional) A `string` value that represents the name of the watermark to add to the PDF document.

If you provide a literal value, type the name of the watermark as configured in document security in administration console. (See Document security administration help

.)

**Exceptions**

This operation can throw an SDKException exception.

## Get all policy names within a policy set operation

Retrieves all policy names within the specified policy set.

For information about the General and Route Evaluation property groups, see Commonoperation properties.

**Input properties**

Properties that specify the source of a list of policy names.

*Policy Set Name*

A `string` value that represents the name of the policy set from which to retrieve the policy names.

If you provide a literal value, type the name of the policy set.

**Output properties**

Properties that specify a list of policy names.

*Result Policy Names*

The list of policy names from the specified policy set. The data type is `list`.

If you provide a variable, select a variable from the Result Policy Names list. Click the plus sign button to display the Variable dialog box to create a variable. (See Creatingvariables).

### Exceptions

This operation can throw an SDKException exception.

## Get all policy set names operation

Retrieves all policy set names.

For information about the General and Route Evaluation property groups, see Commonoperation properties.

### Output properties

Properties that specify the list of policy set names.

*Result Policy Set Names*

(Optional) The list of policy set names. The data type is `list`.

If you provide a variable, select a variable from the Result Policy Names list. Click the plus sign button to display the Variable dialog box to create a variable. (See Creatingvariables).

### Exceptions

This operation can throw an SDKException exception.

## Get license identifier operation

Retrieves the license identifier value for the specified policy-protected document. Use the license identifier to revoke a license on a document or switch a policy.

For information about the General and Route Evaluation property groups, see Commonoperation properties.

### Input properties

Property to specify the document.

*Input PDF Document*

A `document` value that represents the PDF or Microsoft Office document from which to extract the license. The specified document must have been previously secured using document security.

If you provide a literal value, click the ellipsis button to open the Select Asset dialog box. (SeeAbout Select Asset.)

### Output properties

Property to specify the license identifier.

*Result License Identifier*

(Optional) The license identifier associated with a specific document. The data type is `string`.

### Exceptions

This operation can throw an SDKException exception.

## Get policy by policy identifier operation

Retrieves the details of a policy by using its identifier value.

For information about the General and Route Evaluation property groups, see Commonoperation properties.

### Input properties

Property to specify the policy identifier.

*Policy ID*

A `string` value that represents the identifier of the policy to retrieve.

If you provide a literal value, type the identifier.

### Output properties

Property to specify the policy details.

*Result Policy*

(Optional) The retrieved policy details. The data type is `PolicySpec`.

If you provide a variable, select a variable from the Result Policy list. Click the plus sign button to display the Variable dialog box to create a variable. (See Creatingvariables).

### Exceptions

This operation can throw an SDKException exception.

## Inspect Protected Document operation

Retrieves license metadata for the specified policy-protected document.

For information about the General and Route Evaluation property groups, see Commonoperation properties.

### Input properties

Property to specify the document from which to extract license metadata.

*Input RM protected PDF or MS Office Document*

A `document` value that represents the document from which to extract license metadata. This value must represent a PDF or Microsoft Office file secured by document security.

If you supply a literal value, click the ellipsis button to open the Select Asset dialog box. (See About Select Asset.)

### Output properties

Property to specify the license metadata.

*RM Inspect Result*

(Optional) The license metadata. The data type is `RMInspectResult`.

If you provide a variable, select a variable from the RM Inspect Result list. Click the plus sign button to display the Variable dialog box to create a variable. (See Creatingvariables).

### Exceptions

This operation can throw an SDKException exception.

## Protect Document operation

Secures the specified PDF or Microsoft Office document with an existing policy.

For information about the General and Route Evaluation property groups, see Commonoperation properties.

### Input properties

Properties to specify the document to which to apply a policy, the document name, and the policy set. Optionally, you can specify the user profile and domain of the publisher and the document's locale.

*Input PDF or MS Office Document*

A `document` value that represents the document to secure with the policy.

If you supply a literal value, click the ellipsis button to open the Select Asset dialog box. (See About Select Asset.)

*Document Name*

A `string` value that contains the name of the document. This value can have a maximum length of 50 single-byte characters and represents the name of the document audited by the document security service.

If you provide a literal value, type the name of the document.

*Policy Set Name*

(Optional) A `string` value that specifies the policy set to use to secure the document. The value `My Policies` is used if a policy set name is not provided.

If you provide a literal value, type the name of the policy set.

*Policy Name*

A `string` value that specifies the name of the policy to use to secure the document.

If you provide a literal value, type the name of the policy.

*Publisher Username*

(Optional) A `string` value that represents the canonical name of the User Manager user who is the publisher of the document. The Document Publisher controls the policy associated with a document. The Document Publisher's identity is used to apply a policy to a document. It also has special permissions to remove a policy from a document or switch a policy on a document.

To use this property, specify a value for the Publisher Domain property as well.

For long-lived services, both Publisher Username and Publisher Domain values are required.

For short-lived services, if you do not provide values, the user name and domain are picked up from the user account that is used to invoke the process.

If you provide a literal value, type the publisher's user profile.

*Publisher Domain*

(Optional) A `string` value that represents the name of the User Manager domain of the user who is the publisher of the document.

To use this property, specify a value for the Publisher Username property as well.

For long-lived services, specify both Publisher Username and Publisher Domain values.

If you provide a literal value, type the name of the publisher's domain.

*Locale*

(Optional) A `string` value that represents the locale of the policy-protected MS Office file. Set the locale for MS Office documents only to ensure that an appropriate template document is displayed to users. Template documents inform users to install a plug-in. The default value is `en`.

If you provide a literal value, select `en`, `fr`, `ja`, or `de`.

For information about the plug-in, see the Acrobat Reader DC extensions Help.

**Output properties**

Properties that specify the document to which the policy is applied.

*ProtectedDocument*

(Optional) The policy-protected document. The data type is `string`.

If you provide a variable, select a variable from the ProtectedDocument list. Click the plus sign button to display the Variable dialog box to create a variable. (See Creatingvariables).

*PolicyID*

(Optional) The identifier of the policy used to secure the document. The data type is `string`.

If you provide a variable, select a variable from the PolicyID list. Click the plus sign button to display the Variable dialog box to create a variable. (See Creating variables).

*DocumentID*

(Optional) The identifier of the policy-protected document. The data type is `string`.

If you provide a variable, select a variable from the DocumentID list. Click the plus sign button to display the Variable dialog box to create a variable. (See Creating variables).

*MimeType*

(Optional) The MIME type of the policy-protected document. The data type is `string`. Used for MS Office documents only.

If you provide a variable, select a variable from the MimeType list. Click the plus sign button to display the Variable dialog box to create a variable. (See Creating variables).

**Exceptions**

This operation can throw an SDKException exception.

## Remove policy security operation

Removes the security from a policy-protected PDF or Microsoft Office (MS Office) document. The user who invokes this operation must be authenticated with the AEM Forms Server. The user must also be a known publisher of the document or be authorized to remove security from a document.

For information about the General and Route Evaluation property groups, see Commonoperation properties.

**Input properties**

Property to specify the policy-protected document.

*Input PDF Document*

A `document` value that represents the policy-protected PDF or MS Office document.

If you provide a literal value, click the ellipsis button to open the Select Asset dialog box. (See About Select Asset).

### Output properties

Property to specify the unsecured document.

*Result PDF Document*

(Optional) The unsecured PDF or MS Office document. The data type is `document`. If a file with the same name exists in this location, it is overwritten.

If you provide a variable, select a variable from the Result PDF Document list. Click the plus sign button to display the Variable dialog box to create a variable. (See Creatingvariables).

### Exceptions

This operation can throw an SDKException exception.

## Revoke license operation

Revokes the specified license. When you revoke a license, users cannot open the associated PDF document.

You can provide a URL that appears to users when they attempt to open a document whose license is revoked. For example, you can provide a URL that informs users that the document is out of date. Another example is a URL that points to an updated version of the document.

For information about the General and Route Evaluation property groups, see Commonoperation properties.

### Input properties

Properties to specify the license to revoke, the reason, and an optional URL to which a user is directed.

*License Identifier*

A `string` value that identifies the license to revoke.

If you provide a literal value, type the license identifier.

*Reason*

A `string` value that represents the reason why the license was revoked. The default is `DOCUMENT_TERMINATED`.

If you provide a literal value, select one of the following values:

**DOCUMENT_REVISED:**

PDF document was revoked because it was revised.

**DOCUMENT_TERMINATED:**

PDF document was revoked because it no longer exists.

**GENERAL_MESSAGE:**

A generic error occurred while processing the PDF document.

*Revocation URL*

(Optional) A `URL` value that represents the URL that appears to users who attempt to open the document. Type `null` if you do not want to provide a URL.

If you provide a literal value, type the URL.

**Exceptions**

This operation can throw an SDKException exception.

## Switch policy operation

Changes the policy for a specified policy-protected document using its license identifier.

For information about the General and Route Evaluation property groups, see Commonoperation properties.

**Input properties**

Properties to specify the license and new policy.

*License Identifier*

A `string` value that identifies the license of the document to change.

If you provide a literal value, type the license identifier.

*New Policy Set Name*

(Optional) A `string` value that specifies the name of the existing policy set to use for securing the PDF document. The value of `My Policies` is used if a policy set name is not provided.

If you provide a literal value, type the name of the existing policy set.

*New Policy Name*

A `string` value that contains the name of the new policy. This value can have a maximum length of 50 characters.

If you provide a literal value, type the name of the document.

**Exceptions**

This operation can throw an SDKException exception.

## Unlock policy-protected PDF operation

Unlocks a policy-protected document. Unlock a policy-protected document to pass the document to another short-lived service. For example, before the Signature service can sign a policy-protected document, it must be unlocked.

You cannot use this operation to unlock a policy-protected Microsoft Office document.

**NOTE:** When you create long-lived processes, in the administration console, configure the process security as Run As Named User. This configuration is required to unlock a policy-protected PDF. (See Applications and Services Help
.)

For information about the General and Route Evaluation property groups, see Commonoperation properties.

### Input properties

Property to specify the PDF document to unlock.

*Input PDF Document*

A `document` value that represents the policy-protected PDF document.

If you provide a literal value, click the ellipsis button to open the Select Asset dialog box. (See About Select Asset.)

### Output properties

Property to specify the unlocked PDF document.

*Result PDF Document*

(Optional) The unlocked PDF document. The data type is `document`.

If you provide a variable, select a variable from the Result PDF Document list. Click the plus sign button to display the Variable dialog box to create a variable. (See Creatingvariables).

### Exceptions

This operation can throw an SDKException exception.

## Unrevoke license operation

Modifies the state of a revoked license so that it is no longer revoked. When the license is no longer revoked, authorized users can open the associated document.

For information about the General and Route Evaluation property groups, see Commonoperation properties.

### Input properties

Property to specify the identifier of the license to unrevoke.

*License Identifier*

A `string` value that identifies the license to unrevoke.

If you provide a literal value, type the license identifier.

### Exceptions

This operation can throw an SDKException exception.

## Update policy operation

Updates the properties of an existing policy.

For information about the General and Route Evaluation property groups, see Commonoperation properties.

### Input properties

Properties to specify the principals and offline lease period of a policy.

*Policy Set Name*

(Optional) A `string` value that represents the name of the existing policy set that contains the policy to update. If a policy is not specified, the policy is based on the policy of the user who invokes the operation.

If you provide a literal value, type a new name for the policy set.

*Policy Name*

A `string` value that contains the name of the policy. This value can have a maximum length of 50 characters.

If you provide a literal value, type the name of the document.

*Offline Lease Period*

(Optional) A `string` value that represents the number of days a recipient can access a document without a network connection to the server that runs document security. The default is `30`.

If you provide a literal value, type the number of days.

*Principals To Add*

(Optional) A `list` of `Principal Reference` values that represent the principals to add to the policy. If no value is specified, the original principals remain unchanged.

If you provide a literal value, click the plus sign button to display the Add Users and Groups dialog box to add principals. (See AboutAdd Users and Groups). You can also select a principal from the list and click the minus sign button to remove it.

*Principals To Remove*

(Optional) A `list` of `Principal Reference` values that represent the principals to remove.

If you provide a literal value, click the plus sign button to display the Add Users and Groups dialog box to add principals. (See AboutAdd Users and Groups). You can also select a principal from the list and click the minus sign button to remove it.

*Watermark Name*

(Optional) A `string` value that represents the name of the watermark to add to the PDF document.

If you provide a literal value, type the name of the watermark as configured in document security in the administration console. (See Document security

administration help.)

### Exceptions

This operation can throw an SDKException exception.

## Document security exceptions

The document security service provides the following exception for throwing exception events.

### SDKException

Thrown when an error occurs executing a document security service operation for one of the following reasons:

- A parameter that was specified is invalid.
- The license identifier was not specified.
- The license identifier was not revoked.
- The operation was not permitted.
- Could not communicate with the document security service.

## Select users and groups

You can specify individual users as well as groups when defining a list of principals.

### Add users and groups

Use the Add Users and Groups dialog box to select principals to add to a list. For information about searching for users or groups, see Selecta specific user or Selecta specific group.

**A.**

Search by user name or email address

**B.**

Display the search results

**C.**

Filter search results by the specified string

**D.**

Add the selected user or group

**E.**

Display the list of principals to add

**F.**

Remove principal from the Selected User(s) Or Group(s) pane

### Select users

Use the Select User dialog box to add a user to a list of principals.

**A.**

Search by user name or email

**B.**

Filter search results by the specified string

**C.**

Display the search results and select a user to add

**D.**

Add the selected user

### Select groups

Use the Select Group dialog box to add a group to a list of principals.

A. Search by group name or email B. Filter search results by the specified string C. Display the search results and select a group to add D. Add the selected group

## 22.44. Set Value

Sets the value of a data location in the process data model.

For information about using the Set Value service, see Services Reference for AEM Forms

.

### execute operation

Sets the value of one or more data items in the process data model. For example, you can set the value of a process variable, or you can set the value of a form field. Use XPath expressions to specify the data item and the value to set it to.

Variable data is persisted and available through the entire process. You should avoid setting the value of the same variable from different branches in a gateway. If branches in a gateway execute simultaneously and write data to the same variable, one branch may overwrite important data that was saved in the other branch.

**NOTE:** If you are using Acrobat forms and are setting the value of a field in an xfaForm, Document Form, or Form variable, the expression can use a fully-qualified path or a relative path to identify the field.

**NOTE:** To create a node, you must use a fully-qualified path. The path must include the node name fields as the form schema root node.

For information about the General and Route Evaluation property groups, see Commonoperation properties.

#### Mapping properties

For every data item that you want to set the value of, you create a mapping. To create or edit mappings, click the edit button or double-click a row in the Mappings table. This action opens the XPath Builder dialog box. (See XPath Builder(Data Mapping).)

*Location*

The data item in the process data model for which a value is set.

*Expression*

The expression that defines the value for the specified data item.

## 22.45. Signature

The Signature service lets your organization work with digital signatures on PDF documents, such as digitally signing or certifying a PDF document on the AEM Forms Server. To maximize your understanding of the features available with the Signature service, read PDFUtilitiesService. You can perform tasks to help automate tasks accomplished by users using Acrobat or Adobe Reader. Using the Signature service, you can automate many tasks such as digitally signing XDP or PDF files by users.

For example, you can use the Signature service to automatically sign a PDF file with a corporate digital signature. This removes the need for a user to manually sign each PDF document in Acrobat or Adobe Reader.

For information about using the Signature service, see Services ReferenceforAEM Forms

.

## Digital signatures and proxy servers

In some AEM forms environments, users can access the AEM Forms Server either by using a proxy server or by accessing the server directly. For processes that use the Signature service, all of the users must access the AEM Forms Server in the same way: either directly or through the same proxy server.

For example, an external user digitally signs a PDF form and submits it to the proxy server. The process sends the form to an internal user who connects to the AEM Forms Server directly. When the internal user opens the form, the digital signature is invalidated and an error occurs when the user tries to submit the form.

RELATED LINKS:

Signatureservice configuration

Signatureexceptions

AboutSPI Properties

AboutAdd Subject DN

## Signature service configuration

The following service configuration properties can be modified for the Signature service. (See Editingserviceconfigurations.)

Some of the values are used as default values for operation properties. To override the default values, specify different values for the operation properties.

**Execute Document JavaScripts scripts:**

Specifies whether to execute Document JavaScript scripts in Acrobat or Adobe Reader during signature operations. By default, the option is selected, which means to execute Document JavaScript scripts during signature operations.

**Process Documents with Acrobat 9 Compatibility:**

Specifies whether to enable Acrobat 9 compatibility. For example, when this option is selected, Visible Certification in Dynamic PDFs is enabled. By default, the option is selected, which means to allow for Acrobat 9 compatibility.

**Embed Revocation Info While Signing:**

Specifies whether revocation information is embedded while signing the PDF document. By default, the option is selected, which means that revocation information is embedded when the Signature service operation signs the PDF document.

**Embed Revocation Info While Certifying:**

Specifies whether the revocation information is embedded while certifying the PDF document. By default, the option is selected, which means that revocation information is embedded when the Signature service signs the PDF document.

**Enforce Embedding of Revocation Info for all Certificates During Signing/Certification:**

Specifies whether a signing or certification operation fails if valid revocation information for all certificates is not embedded. When a certificate does not contain any Certificate Revocation List (CRL) or Online Certification Server Protocol (OCSP) information, it is considered valid, even if no revocation information is retrieved. By default, the option is not selected, which means that signing and certification operations do not fail regardless of whether revocation information is embedded.

**Revocation Check Order:**

Specifies the order of mechanisms to use to perform revocation checking. By default, the selected value is OCSPFirst. Select one of the following values:

- **CRLFirst:** Use Certificate Revocation List (CRL) before Online Certificate Status Protocol (OCSP).
- **OCSPFirst:** Use OCSP before CRL.

**Maximum Size of Revocation Archival Info:**

Specifies the maximum size of the revocation archival info in kilobytes. AEM Forms attempts to store as much revocation information as possible without exceeding the limit. The default value is `10` KB.

**Verification Time Option:**

Specifies the time of verification of a signer's certificate. By default, the selected value is Signing Time. Select one of the following values:

- **Signing Time:** The time that the signature was applied as given by the signer's computer.
- **Current Time:** The time that the verification operation is being carried out.
- **Secure Time Else Current Time:** The time specified by a trusted time-stamping authority.

**Use Revocation Information Archived in Signature During Validation:**

Specifies whether the revocation information that is archived with the signature is used for revocation checking. By default, the option is selected.

**Use Validation Information Stored in the Document for Validation of Signatures:**

Specifies whether to use the validation information that is stored in the PDF document to validate digital signatures. This option is a part of the Long Term Validation support available in Acrobat 9.1 and the Signature service, which creates a Digital Signature Standard (DSS) dictionary in the PDF document. The DSS dictionary stores the validation information for the signatures in the document. The validation information includes certificates, revocation information, and timestamp information. In previous releases of Acrobat and AEM Forms, the validation information was stored as a part of the digital signature.

The option is selected by default, which means to use the validation information that is stored in the PDF document to validate digital signatures.

**Maximum Nested Verification Sessions Allowed:**

Specifies the maximum number of nested verification sessions that are allowed. The AEM Forms Server uses this value to prevent an infinite loop. Infinite loops can occur while verifying the OCSP or CRL signer certificates when the OCSP or CRL is not set up correctly. The default value is 5.

**Maximum Clock Skew for Verification:**

Specifies the maximum time, in minutes, that the signing time can be after the validation time. If the clock skew is greater than this value, the signature is not valid. The default value is 65 min.

**Certificate Lifetime Cache (In Minutes):**

Specifies the lifetime of a certificate, retrieved online or through other means, in the cache. The default value is 1440 min.

**Transport Options**

**Proxy Host:**

Specifies the URL of the proxy host. A proxy host is only used when some valid value is provided.

**Proxy Port:**

Specifies the port to use for the proxy. Valid port number values are 0 – 65535. The default value is 80.

**Proxy Login Username:**

Specifies the user name to use to log in to the proxy host. This option is used when a valid Proxy Host and Proxy Port are configured.

**Proxy Login Password:**

Specifies the password to use to log in to the proxy host. This option is used when a valid Proxy Host and Proxy Port are configured.

**Maximum Download Limit:**

Specifies the maximum amount of data, in megabytes, that can be received per connection. Valid download limit values are 1 MB to 1024 MB. The default value is 16 MB.

**Connection Time Out:**

Specifies the maximum time to wait, in seconds, for establishing a new connection. Valid time-out values are 1 – 300 sec. The default value is 5.

**Socket Time Out:**

Specifies maximum time to wait, in seconds, before a socket time-out (while waiting for data transfer) occurs. Valid time-out values are 1  3600– sec. The default value is 30 sec.

### Path Validation Options

**Require Explicit Policy:**

Specifies whether the path must be valid for at least one of the certificate policies that is associated with the trust anchor of the signer certificate. By default, this value is not selected, which means that no certificate policy is required to be associated with the trust anchor of the signer certificate.

**Inhibit ANY Policy:**

Specifies whether the policy object identifier (OID) must be processed if it is included in a certificate. By default, the option is not selected, which means that the OID does not need to be processed.

**Inhibit Policy Mapping:**

Specifies whether policy mapping is allowed in the certification path. By default, the option is not selected, which means that policy mapping is not allowed in the certification path.

**Check All Paths:**

Specifies whether all paths must be validated or validation stops after finding the first valid path. By default, the option is deselected, which means that validation stops after the first valid path.

**LDAP Server:**

Specifies the URL or path of the LDAP server used to look up certificates for path validation. For example, you can type `www.ldap.com` for the URL or `ldap://ssl.ldap.com:200` for the path and port.

**Follow URIs in Certificate AIA:**

Specifies whether Uniform Resource Identifiers (URIs) in Certificate Authority Information Access (AIA) are processed during path discovery. By default value, the option is not selected, which means that during path discovery, do not process URIs in the AIA.

**Basic Constraints Extension Required in CA Certificates:**

Specifies whether the certificate authority (CA) Basic Constraints certificate extension must be present for CA certificates. Some early German certified root certificates (7 and earlier) are not compliant to RFC 3280

and do not contain the basic constraint extension. If it is known that a user's EE certificate chains up to such a German root, deselect this option. By default, the option is selected, which means that the CA Basic Constraints certificate must be present.

**Require Valid Certificate Signature During Chain Building:**

Specifies whether the chain builder requires valid signatures on certificates used to build chains. When this option is selected, the chain builder does not build chains with invalid Digital Signature Algorithm (DSA) signatures on certificates. For example, in a chain CA > ICA > EE where the signature for EE is not valid, the chain building stops at ICA. EEs are not included in the chain. This setting

does not affect DSA signatures. By default, the option is deselected, which means the full three-certificate chain is produced.

**Timestamp Provider Options**

**TSP Server URL:**

Specifies the URL of the default timestamp provider. This option is not used when no value is provided.

**TSP Server Username:**

Specifies user name to use to access the timestamp provider. This option is used when a value is provided for the TSP Server URL option.

**TSP Server Password:**

Specifies the password to use to access the timestamp server. This option is used when a value is provided for the TSP Server Username and TSP Server URL options.

**Request Hash Algorithm:**

Specifies the hashing algorithm to use while creating the request for the timestamp provider. Select one of the following values:

- **SHA1:** (Default) The Secure Hash Algorithm that has a 160-bit hash value.
- **SHA256:** The Secure Hash Algorithm that has a 256-bit hash value.
- **SHA384:**The Secure Hash Algorithm that has a 384-bit hash value.
- **SHA512:** The Secure Hash Algorithm that has a 512-bit hash value.
- **RIPEMD160:** The RACE Integrity Primitives Evaluation Message Digest that has a 160-bit message digest algorithm and is not FIPS-compliant.

**Revocation Check Style:**

Specifies the revocation-checking style used for determining the trust status of the timestamp provider's certificate from its observed revocation status. Select one of the following values:

- **NoCheck:** Does not check for revocation.
- **BestEffort:** Checks for revocation of all certificates when possible.
- **CheckIfAvailable:** (Default) Checks for revocation of all certificates only when revocation information is available.
- **AlwaysCheck:** Checks for revocation of all certificates.

**Send Nonce:**

Specifies whether a nonce is sent with the request. A *nonce* is a parameter that varies with time. These parameters can be a timestamp, a visit counter on a web page, or a special marker. The parameters are intended to limit or prevent the unauthorized replay or reproduction of a file. By default, the option is selected, which means that a nonce is sent with the request.

**Use Expired Timestamps During Validation:**

Specifies whether to use a timestamp that has expired. The default is selected, which means to use the time present in expired timestamps during validation of the signature.

**TSP Response Size:**

Specifies the estimated size, in bytes, of the timestamp server (TSP) response. This value represents the maximum size of the timestamp response that the configured timestamp provider can return. Configuring an undersized value can cause the operation to fail and errors to be seen the server logs; however, configuring an oversized value causes the size to be larger than necessary. It is recommended that this value is not modified unless the timestamp server requires a response size to be less than 4096 bytes. Do not change this value unless you are certain what to change the value to. Valid response sizes are `60B` to `10240B`. The default value is `4096B`.

**Ignore TimeStamp Server Extension**

Select the **Ignore TimeStamp Server Extension** option to stop AEM Forms server from contacting the specified time stamp server. Selecting the option helps avoid process failures which happen due to connection timeout between AEM Forms and time stamp servers.

**Certificate Revocation List Options**

**Consult Local URI First:**

Specifies whether the Certificate Revocation List (CRL) location is provided in Local URI for CRL Lookup. The Local URI must have preference over any location specified within a certificate for revocation checking. By default, the option is deselected, which means the locations are specified in the certificate before using the local URI.

**Local URI for CRL Lookup:**

Specifies the URL of the local CRL provider. This value is only used when the Consult Local URI First setting is selected.

**Revocation Check Style:**

Specifies the revocation-checking style used for determining the trust status of the CRL provider's certificate from its observed revocation status. Select one of the following values:

- **NoCheck:** Does not check for revocation.
- **BestEffort:** (Default) Checks for revocation of all certificates when possible.
- **CheckIfAvailable:** Checks for revocation of all certificates only when revocation information is available.
- **AlwaysCheck:** Checks for revocation of all certificates.

**LDAP Server for CRL Lookup:**

Specifies the URL or path of the Lightweight Directory Access Protocol (LDAP) server used to retrieve information about the certificate revocation list (CRL). The LDAP server searches for CRL information using the distinguished name (DN) according to the rules specified in RFC3280

, section 4.2.1.14. For example, you can type `www.ldap.com` for the URL or `ldap://ssl.ldap.com:200` for the path and port.

**Go Online:**

Specifies whether to access the network to retrieve CRL information. CRL information is cached for optimal usage of the network. When the option is deselected, it means not to go online. By default, the option is selected, which means to access the network.

**Ignore Validity Dates:**

Specifies whether to ignore the response's thisUpdate and nextUpdate times, which prevents any negative effect times have on response validity. The thisUpdate and nextUpdate times are external sources that are retrieved through HTTP or LDAP and can be different for each revocation information. When this option is selected, it means to ignore the thisUpdate and nextUpdate times. By default, this option is deselected, which means to use the thisUpdate and nextUpdate times.

**Require AKI Extension in CRL:**

Specifies whether the Authority Key Identifier (AKI) extension must be present in the CRL. The AKI extension can be used for CRL validation. When the option is selected, it means that the AKI extension must be present. By default, the option is deselected, which means that the AKI extension does not have to be present.

**Online Certificate Status Protocol Options**

**OCSP Server URL:**

Specifies the local URL, which is the location of the Online Certificate Status Protocol (OCSP) server, which is the location of the configured OCSP server. The value is only used when the LocalURL or UseAIAIfPresentElseLocal values are in URL To Consult Option.

**URL To Consult Option:**

Specifies the list and order of the OCSP servers used to perform the revocation check. Select one of the following values:

- **UseAIAInCert:** Use the URL of an online certificate status protocol server specified in the Authority Information Access (AIA) extension in the certificate.
- **LocalURL:** Use the specified URL for the OCSP server specified in the OCSP Server URL option.
- **UseAIAIfPresentElseLocal:** Use the URL of the OCSP server specified in the AIA extension in the certificate if present. If the certificate is not present, use the URL configured in the OCSP Server URL.
- **UseAIAInSignerCert:** (Default) Use the URL of the OCSP server specified in the AIA extension in the OCSP request of the signer certificate.

**Revocation Check Style:**

Specifies the revocation-checking style used for verifying the trust status of the CRL provider's certificate from its observed revocation status. Select one of the following values:

- **NoCheck:** Does not check for revocation.
- **BestEffort:** Checks for revocation of all certificates when possible.
- **CheckIfAvailable:** (Default) Checks for revocation of all certificates only when revocation information is available.
- **AlwaysCheck:** Checks for revocation of all certificates.

**Send Nonce:**

Specifies whether a nonce is sent with the request. A *nonce* is a parameter that varies with time. These parameters can be a timestamp, a visit counter on a web page, or a special marker. The parameters are intended to limit or prevent the unauthorized replay or reproduction of a file. When the option is deselected, a nonce is not sent with the request. By default, the option is selected, which means a nonce is sent with the request.

**Max Clock Skew Time:**

Specifies the maximum allowed skew, in minutes, between response time and local time. The minimum value is `0` and the maximum value is `2147483647` min. The default value is `5` min.

**Response Freshness Time:**

Specifies the maximum time, in minutes, for which a preconstructed OCSP response is considered valid. Valid response freshness times are `1` – `2147483647` min. The default value is `525600` min. (one year).

**Sign OCSP Request:**

Specifies whether the OCSP request must be signed. When the option is selected, it means that the OCSP request must be signed. By default, the option is deselected, meaning that the OCSP request is not required to be signed.

**Request Signer Credential Alias:**

Specifies the trust store credential alias to use for signing the OCSP request if signing is enabled. The alias is used if the Sign OCSP Request option is selected.

**Go Online:**

Specifies whether to access the network to retrieve OCSP information. Embedded and cached OCSP responses are used on the server to reduce the amount of network traffic generated due to OCSP checking. When the option is deselected, OCSP information is not retrieved from the network and only embedded and cached OCSP information is used. By default, the option is selected, which means to access the network for OCSP information.

**Ignore the Response's thisUpdate and nextUpdate Times:**

Specifies whether to ignore the response's thisUpdate and nextUpdate times, which prevents any negative effect times have on response validity. The thisUpdate and nextUpdate times are retrieved from external sources by using HTTP or LDAP and can be different for each revocation information. When the option is selected, it means to ignore the thisUpdate and nextUpdate times. By default, the option is deselected, which means to use the thisUpdate and nextUpdate times.

**Allow OCSPNoCheck Extension:**

Specifies whether the OCSPNoCheck extension is allowed in the response signing certificate. An OCSPNoCheck extension can be present in the OCSP Responder's certificate to prevent infinite loops from occurring during the validation process. When the option is deselected, it means that the OCSPNoCheck extension is not allowed. By default, the option is selected, which means the OCSPNoCheck extension is allowed.

**Require OCSP ISIS-MTT CertHash Extension:**

Specifies whether a certificate public key hash extension must be present in OCSP responses. This extension is required for SigQ validation. SigQ compliance requires the CertHash extension to be in the OCSP responder certificate. Select this option when processing for SigQ compliance and supported OCSP responders. When the option is selected, it means that the certificate public key hash extension must be present. By default, the option is deselected, which means that the presence of a certificate public key extensions is not required.

**Error Handling Options for Debugging**

**Purge Certificate Cache on Next API Call:**

Use this option for debugging purposes in a non-production environment. Specifies whether to purge the certificate cache when the next Signature service operation executes. When the option is selected, it means that the certificate cache on the AEM Forms Server is purged. By default, the option is deselected, which means that the certificate cache is not purged. After the first Signature operation executes, the option becomes deselected.

**Purge CRL Cache on Next API Call:**

Use this option for debugging purposes in a non-production environment. Specifies whether to purge the Certificate Revocation List (CRL) cache when the next Signature service operation executes. When the option is selected, it means that the CRL Cache on the AEM Forms Server is purged. By default, the option is deselected, which means that the CRL cache on the AEM Forms Server is not purged. After the first Signature operation executes, the option becomes deselected.

**Purge OCSP Cache on Next API Call:**

Use this option for debugging purposes in a non-production environment. Specifies whether to purge the Online Certification Server Protocol (OCSP) cache when the next Signature service operation executes. When the option is selected, it means that the OCSP Cache on the AEM Forms Server is purged. By default, the option is deselected, which means that the OCSP cache is not purged. After the first Signature operation executes, the option becomes deselected.

## Add Invisible Signature Field operation

Creates an invisible signature field in a PDF document. Use an invisible signature field when you do not want the signature field displayed to a user.

For example, your application must use the AEM Forms Server to digitally sign a PDF form but the form has no signature field on it. You use the Add Invisible Signature Field operation to add a hidden signature

field to the PDF form. An invisible signature field is useful when you sign the signature field on the server. It is also useful when do not want a user to see the digital signature field in the form, however, the user can still see that an invisible signature exists in the Signatures tab in Acrobat or Adobe Reader. After you add the signature field, you can sign the signature field using the SignSignatureField or CertifyPDF operation in subsequent steps in the process.

**IMPORTANT:** You cannot add an invisible signature field to a dynamic PDF form.

For information about the General and Route Evaluation property groups, see Commonoperationproperties.

Some properties of this operation provide the following buttons to manage entries in lists:

**Add A List Entry:**

Adds an entry to the list. Depending on the option, you type the information, select an item from a drop-down list, or select a file from a network location or computer. When you select a file from a location on your computer, during run time, the file must exist in the same location on the AEM Forms Server.

**Delete Selected List Entry:**

Removes an entry from the list.

**Move Selected List Entry Up One Row:**

Moves the selected entry up in the list.

**Move Selected List Entry Down One Row:**

Moves the selected entry down in the list.

### Common properties

Properties to specify mandatory values.

*Input PDF*

A `document` value that represents the PDF document to which the invisible signature field is added.

If you provide a literal value, clicking the ellipsis button opens the Select Asset dialog box. (See AboutSelectAsset.)

*Signature Field Name*

A `string` value for the name of the signature field to add. The fully qualified name of the signature field must be specified.

### Advanced properties

Properties to specify the PDF document fields that are locked after the signature field is signed, the seed value dictionary associated with the digital signature.

*Field MDP Options Spec*

(Optional) A `FieldMDPOptionSpec` value that specifies the PDF document fields that are locked after the signature field is signed.

If you provide a literal value, you can set the following options.

**Field Locking Action:**

A list that sets the type of action to use to lock fields in a PDF document. Select one of these values:

- **All Fields:** Lock all fields in the PDF document.
- **Include Fields:** Lock only the fields specified in the Application To Form Fields option.
- **Exclude Fields:** Lock all fields except for those specified in the Applicable To Form Fields option.

**Applicable to Form Fields:**

Sets a comma-separated list of fully qualified field names that indicate which fields the action is applicable or not applicable to. This option is available when the Field Locking Action option is set to a value of Include Fields or Exclude Fields.

*Seed Value Options Spec*

(Optional) A `PDFSeedValueOptionSpec` value that represents the seed value dictionary that is associated with a signature field. A seed value dictionary contains entries that constrain information that is used at the time the signature is applied. The options are used for specifying the document signature settings.

If you provide a literal value, you can set the following options:

**Signature Handler Options:**

Options for specifying the filters and subfilters that are used for validating a signature field. The signature field is embedded in a PDF document and the seed value dictionary is associated with a signature field.

- **Signature Handler:** A list of handlers to use for the digital signatures. Adobe.PPKLite is a valid value that can be selected to represent the creation and validation of Adobe-specific signatures. You can use other signature handlers by typing string values, such as `Entrust.PPEF`, `CIC.SignIt`, and `VeriSign.PPKVS`. For information about supported signature handlers, see *PDF Utilities Service.* No default value is selected. The following signature handler is available to be selected from the list:
  - **Adobe.PPKLite:** The recommended handler for signing PDF documents.

    ***Required:*** *Select to specify that the signature handler is used for the seed value. It is not selected by default.*

- **Signature SubFilter:**The supported subfilter names, which describe the encoding of the signature value and key information. Signature handlers must support the listed subfilters; otherwise, the signing fails. These string values, which you must type, are valid for public-key cryptographic (see *PDF Utilities Service*):
  - **adbe.x509.rsa_sha1:**The key contains a DER-encoded PKCS#1 binary data object. The binary objects represent the signature that is obtained as the RSA encryption of the byte range

SHA-1 digest with the private key of the signer. Use this value when signing PDF documents using PKCS#1 signatures.

– **adbe.pkcs7.detached:**The key is a DER-encoded PKCS#7 binary data object that contains the signature. No data is encapsulated in the PKCS#7-signed data field.
– **adbe.pkcs7.sha1:** The key is a DER-encoded PKCS#7 binary object that represents the signature value. The SHA-1 digest of the byte range digest is encapsulated in the PKCS#7 signed data.

***Required:*** *Select to specify that signature subfilters are used for the seed value. It is not selected by default.*

- **Digest Methods:** The list of acceptable hashing algorithms to use. Add an item to the list and select an encryption algorithm. Select one of these values:
  – **SHA1:** (Default) The Secure Hash Algorithm that has a 160-bit hash value.
  – **SHA256:** The Secure Hash Algorithm that has a 256-bit hash value.
  – **SHA384:** The Secure Hash Algorithm that has a 384 bit-hash value.
  – **SHA512:** The Secure Hash Algorithm that has a 512 bit-hash value.
  – **RIPEMD160:**The RACE Integrity Primitives Evaluation Message Digest that has a 160-bit message digest algorithm and is not FIPS-compliant.

  ***Required:*** *Select to specify that the signature encryption algorithms are used for the seed value. It is not selected by default.*

- **Minimum Signature Compatibility Level:** The minimum PDF version to use to sign the signature field. Select one of these values:
  – **PDF 1.5:** Use PDF Version 1.5.
  – **PDF 1.7:** Use PDF Version 1.7.

  ***Required:*** *Select to specify the minimum signature compatibility level is used for the seed value. It is not selected by default.*

**Signature Information:**

A group of options for specifying the reasons, timestamp, and details of the digital signature.

- **Include Revocation Information in Signature:** Select to specify that revocation information must be embedded as part of the signature for long-term validation support. When you deselect this option, the revocation information is not embedded as part of the signature. By default, this option is deselected.

  ***Required:*** *Select to specify that revocation checking is required for the seed value. It is not selected by default.*

- **Signing Reasons:** The list of reasons that are associated with the seed value dictionary used for signing the PDF document. Add an item to the list and type a reason.

  ***Required:*** *Select to specify that the associated reasons are included for the seed value. It is not selected by default.*

- **TimeStamp Server URL:** The URL that specifies the location of the timestamp server to use when signing a PDF document.

***Required:*** *Select to specify that the timestamp server is required for the seed value. It is not selected by default.*

- **Signing/Enrollment Server URL:** The location of the server that provides a web service. The web service digitally signs a PDF document or enrolls for new credentials.

  ***Required:*** *Select to specify that the signing or enrollment server is used for the seed value. It is not selected by default.*

- **Server Type:** The type of server to use for the value specified for the Signing/Enrollment Server URL option. Select one of these values:
  - **Browser:**(Default) The URL references content that is displayed in a web browser to allow enrolling for a new credential if a matching credential is not found.
  - **ASSP:** The URL references a signature web service. The web service is used to digitally sign the PDF document on a server. The server is specified in the Signing/Enrollment Server URL option in this operation.

  ***Required****: Select to use the web service to sign the PDF document. It is not selected by default.*

**Signature Type:**

The changes that are permitted after the signature is added and legal attestations are provided.

- **Type of Signature:** The list representing the type of signatures that can be applied to the signature field. Select one of these values:

  ***Any:*** *(Default) Any type of signature can be applied when filling forms, instantiating page templates, or creating, deleting, and modifying annotations.*

  ***Recipient Signature:*** *Restricts the signer to apply a Four Corner security model on the signature field.*

  ***Certification Signature:*** *Constrains the signer to apply a certification signature on the signature field with specified permissions. The specified permissions are configured in the Field MDP Options Spec property for this operation. Select one of these values:*
  - **No changes allowed**: The end user is not permitted to change the form. Any change invalidates the signature.
  - **Form fill-in and digital signatures**: The end user is permitted to fill the form, instantiate page templates, and sign the form.
  - **Annotations, form fill-in, and digital signatures**: The end user is permitted to fill the form, instantiate page templates, sign the form, and create annotations, deletions and modifications.

- **Legal Attestations:** The list of legal attestations that are associated with the seed value. Legal attestation constraints affect only a certification signature. When you select Any or Certificate Signature option for the Type of Signature, you can add a legal attestation to the list by typing it.

  ***Required:*** *Select to specify that legal attestations are used for the seed value. It is not selected by default.*

**Signing Certificates:**

The list of certificates, keys, issuers, and policies that are used for a digital signature. Add certificates, keys, issuers, and policies to the list by using the Open dialog box.

- **Signing Certificates:** A list of certificates that are used for certifying and verifying a signature.

  ***Required:*** *Select to specify that signing certificates are used for the seed value. It is not selected by default.*
- **Subject Distinguished Name:** The list of dictionaries, where each dictionary contains key value pairs that specify the subject distinguished name (DN). The DN must be present within the certificate for it to be acceptable for signing. Add DNs to the list by using the Add Subject DN dialog box. (See AboutAddSubject DN.)

  ***Required:*** *Select to specify that subject distinguished names are used for the seed value. It is not selected by default.*
- **KeyUsage:** The list of key usage extensions that must be present for signing a certificate. Add an entry to the list and select the key usage. Additional key usage entries are available in *PDF Utilities Service*. Select one of these key usage values for each entry:
  - **Don't Care:**(Default) The key usage extension is optional.
  - **Require Key Usage:** The key usage extension must be present.
  - **Exclude Key Usage:** The key usage extension must not be present.

  ***Required:*** *Select to specify that key usage extensions are used for the seed value. It is not selected by default.*

**Issuers and Policies:**

The list of certificate issuers, policies, and associated object identifiers.

- **Certificate Issuers:** The list of certificate issuers. Add certificate issuers to the list by using the Open dialog box.

  ***Required:*** *Select to specify that certificate issuers are used for the seed value. It is not selected by default.*
- **Certificate Policies And Associated Object Identifiers:** The list of certificate policies that are associated with the certificate seed value. Add certificate policies to the list by typing it.

  ***Required:*** *Select to specify that certificate policies and associated identifies are used for the seed value. It is not selected by default.*

**Output properties**

Property to specify the output PDF document.

*Output PDF*

The location in the process data model to store the PDF document. The PDF document has an invisible signature field added to it. The data type is `document`.

### Exceptions

This operation can throw PDFOperationException, PermissionsException, InvalidArgumentException, and DuplicateSignatureFieldException exceptions.

## Add Signature Validation Information operation

Adds signature validation information for long-term validation of digital signatures applied to a PDF document. When a digital signature expires, you can no longer verify the PDF document. Long-term signature validation allows for the validity of a PDF document to be extended when the signature information cannot be verified. Signature information cannot be verified when the signature information expires, the resources to validate the signature are not available, or the certificate used to sign it is revoked. Use this operation to add validation information (required certificates, CRLs, and OCSP responses) to digitally signed PDF documents to extend them for long-term validation.

For example, your application requires that the validation period for a signed PDF document is extended. To extend the validation period, add updated validation information to the PDF document so that it remains valid. Use the Add Signature Validation Information to update the validation information in the signed PDF document.

For information about the General and Route Evaluation property groups, see Commonoperationproperties.

### Common properties

Properties to specify signed PDF document and signature field to extend for long-term validation.

*Input PDF*

A `document` value that represents the PDF document that the invisible signature field is added to.

If you provide a literal value, clicking the ellipsis button opens the Select Asset dialog box. (See AboutSelectAsset.)

*Signature Field Name*

A `string` value for the name of the signature field to add. The fully qualified name of the signature field must be specified.

### Signature Validation Options properties

Properties for updating validation information to extend a signed PDF document for long-term validation.

*OCSP Options Spec*

(Optional) An `OCSPOptionSpec` value that represents settings for using Online Certificate Status Protocol (OCSP) revocation checking. To provide a literal value, specify the following options.

**URL to Consult Option:** Sets the list and order of the OCSP servers used to perform the revocation check. Select one of these values:

**UseAIAInCert:**

(Default) Use the URL of an online certificate status protocol server specified in the Authority Information Access (AIA) extension in the certificate. The AIA extension is used to identify how to access certificate authority (CA) information and services for the issuer of the certificate.

**LocalURL:**

Use the specified URL for the OCSP server specified in the OCSP Server URL option.

**UseAIAIfPresentElseLocal:**

Use the URL of the OCSP server specified in the AIA extension in the certificate if present. If the AIA extension is not present in the certificate, use the URL that is configured in the OCSP Server URL.

**UseAIAInSignerCert:**

Use the URL of the OCSP server specified in the AIA extension in the OCSP request of the signer certificate.

**OCSP Server URL:**

Sets the URL of the configured OCSP server. The value is used only when the LocalURL or UseAIAIf-PresentElseLocal values are in URL To Consult Option.

**Revocation Check Style:**

Sets the revocation-checking style that is used for verifying the trust status of the CRL provider's certificate from its observed revocation status. Select one of these values:

Sets the URL of the configured OCSP server. The value is used only when the LocalURL or UseAIAIf-PresentElseLocal values are in URL To Consult Option.

Revocation Check Style:

Sets the revocation-checking style that is used for verifying the trust status of the CRL provider's certificate from its observed revocation status. Select one of these values:

**NoCheck:**

Does not check for revocation.

**BestEffort:**

Checks for revocation of all certificates when possible.

**CheckIfAvailable:**

(Default) Checks for revocation of all certificates only when revocation information is available.

**AlwaysCheck:**

Checks for revocation of all certificates.

**Max Clock Skew Time (Minutes):**

Sets the maximum allowed skew, in minutes, between response time and local time. Valid skew times are `0` - `2147483647` min. The default value is `5` min.

**Response Freshness Time (Minutes):**

Sets the maximum time, in minutes, for which a preconstructed OCSP response is considered valid. Valid response freshness times are `1`- `2147483647` min. The default value is `525600` min. (one year).

**Send Nonce:**

Select this option to send a nonce with the OCSP request. A *nonce* is a parameter that varies with time. These parameters can be a timestamp, a visit counter on a web page, or a special marker. The parameter is intended to limit or prevent the unauthorized replay or reproduction of a file. When the option deselected, a nonce is not sent with the request. By default, the option is selected.

**Sign OCSP Request:**

Select this option to specify that the OCSP request must be signed. When the option is deselected, the OCSP request does not need be signed. By default, the option deselected.

**Request Signer Credential Alias:**

Sets the credential alias used for signing the OCSP request when signing is enabled.

**Go Online for OCSP:**

Select this option to access the network for OCSP information. The network can be accessed to retrieve OCSP information for OCSP checking. The AEM Forms Server uses embedded and cached OCSP information when possible to reduce the amount of network traffic generated due to OCSP checking. When the option is deselected, OCSP checking is not retrieved from the network, and only embedded and cached OCSP information is used. By default, the option is selected.

**Ignore Validity Dates:**

Select this option to use the OCSP response thisUpdate and nextUpdate times. Ignoring these response times prevents any negative effect on response validity. The thisUpdate and nextUpdate times are retrieved from external sources by using HTTP or LDAP, and can be different for each revocation information. When the option is deselected, the thisUpdate and nextUpdate times are ignored. By default, the option is deselected.

**Allow OCSP NoCheck Extension:**

Select this option to allow an OCSPNoCheck extension in the response signing certificate. An OCSP-NoCheck extension can be present in the OCSP Responder’s certificate to prevent infinite loops from occurring during the validation process. When the option is deselected, the OCSPNoCheck extension is not used. By default, the option is selected.

**Require OCSP ISIS-MTT CertHash Extension:**

Select this option to specify that certificate public key hash (CertHash) extensions must be present in OCSP responses. This extension is required for SigQ validation. SigQ compliance requires the

CertHash extension to be in the OCSP responder certificate. Select this option when processing for SigQ compliance and supported OCSP responders. When the option is deselected, the CertHash extension presence in the OCSP response is not required. By default, the option is deselected.

Sets the maximum allowed skew, in minutes, between response time and local time. Valid skew times are `0` - `2147483647` min. The default value is `5` min.

Response Freshness Time (Minutes):

Sets the maximum time, in minutes, for which a preconstructed OCSP response is considered valid. Valid response freshness times are `1`- `2147483647` min. The default value is `525600` min. (one year).

Send Nonce:

Select this option to send a nonce with the OCSP request. A *nonce* is a parameter that varies with time. These parameters can be a timestamp, a visit counter on a web page, or a special marker. The parameter is intended to limit or prevent the unauthorized replay or reproduction of a file. When the option deselected, a nonce is not sent with the request. By default, the option is selected.

Sign OCSP Request:

Select this option to specify that the OCSP request must be signed. When the option is deselected, the OCSP request does not need be signed. By default, the option deselected.

Request Signer Credential Alias:

Sets the credential alias used for signing the OCSP request when signing is enabled.

Go Online for OCSP:

Select this option to access the network for OCSP information. The network can be accessed to retrieve OCSP information for OCSP checking. The AEM Forms Server uses embedded and cached OCSP information when possible to reduce the amount of network traffic generated due to OCSP checking. When the option is deselected, OCSP checking is not retrieved from the network, and only embedded and cached OCSP information is used. By default, the option is selected.

Ignore Validity Dates:

Select this option to use the OCSP response thisUpdate and nextUpdate times. Ignoring these response times prevents any negative effect on response validity. The thisUpdate and nextUpdate times are retrieved from external sources by using HTTP or LDAP, and can be different for each revocation information. When the option is deselected, the thisUpdate and nextUpdate times are ignored. By default, the option is deselected.

Allow OCSP NoCheck Extension:

Select this option to allow an OCSPNoCheck extension in the response signing certificate. An OCSP-NoCheck extension can be present in the OCSP Responder’s certificate to prevent infinite loops from occurring during the validation process. When the option is deselected, the OCSPNoCheck extension is not used. By default, the option is selected.

Require OCSP ISIS-MTT CertHash Extension:

Select this option to specify that certificate public key hash (CertHash) extensions must be present in OCSP responses. This extension is required for SigQ validation. SigQ compliance requires the CertHash extension to be in the OCSP responder certificate. Select this option when processing for SigQ compliance and supported OCSP responders. When the option is deselected, the CertHash extension presence in the OCSP response is not required. By default, the option is deselected.

*CRL Options Spec*

(Optional) A `CRLOptionSpec` value that represents the certificate revocation list (CRL) preferences when CRL is used to perform revocation checking. If you provide a literal value, specify the following options.

**Consult Local URI First:**

Select this option to use the CRL location provided as a local URI before any specified locations within a certificate. The CRL location provided is used for revocation checking. When this option is selected, it means the local URI is used first. When this option is deselected, the locations specified in the certificate before using the local URI are used. By default, the option is deselected.

**Local URI for CRL Lookup:**

Sets the URL for the local CRL store. This value is used only if the Consult Local URI First option is selected. No default value is provided.

**Revocation Check Style:**

Sets the revocation-checking style used for verifying the trust status of the CRL provider's certificate from its observed revocation status. Select one of these values:

- **NoCheck:** Does not check for revocation.
- **BestEffort:** (Default) Checks for revocation of all certificates when possible.
- **CheckIfAvailable:** Checks for revocation of all certificates only when revocation information is available.
- **AlwaysCheck:** Checks for revocation of all certificates.

**LDAP Server:**

Sets the URL or path of the Lightweight Directory Access Protocol (LDAP) server used to retrieve information about the certificate revocation list (CRL). The LDAP server searches for CRL information using the distinguished name (DN) according to the rules specified in RFC 3280

, section 4.2.1.14. For example, you can type `www.ldap.com` for the URL or `ldap://ssl.ldap.com:200` for the path and port. No default value is provided.

**Go Online for CRL Retrieval:**

Select this option to access the network to retrieve CRL information. CRL information is cached on the server to improve network performance. CRL information is retrieved online only when necessary. When this option is deselected, CRL information is not retrieved online. By default, the option is selected.

**Ignore Validity Dates:**

Select this option to use thisUpdate and nextUpdate times. Ignoring the response's thisUpdate and nextUpdate times prevents any negative effect on response validity. The thisUpdate and nextUpdate times are retrieved from external sources by using HTTP or LDAP and can be different for each revocation information. When the option is deselected, the thisUpdate and nextUpdate time are ignored. By default, the option deselected.

**Require AKI Extension in CRL:**

Select this option to specify that the Authority Key Identifier (AKI) extension must be present in the CRL. The AKI extension can be used for CRL validation. When this option is deselected, the presence of the AKI extension the CRL is not required. By default, the option is deselected.

*Path Validation Options Spec*

(Optional) A `PathValidationOptionSpec` that represents the settings that control RFC3280-related path validation options. For example, you can indicate whether policy mapping is allowed in the certification path. (See RFC3280

). If you provide a literal value, you can set the following options.

**Require Explicit Policy:**

Select this option to specify that the path must be valid for at least one of the certificate policies in the user initial policy set. When this option is deselected, the path validity is not required. By default, the option is deselected.

**Inhibit Any Policy:**

Select this option to specify that a policy object identifier (OID) must be processed if it is included in a certificate. When deselected, any policy can be selected. By default, the option is deselected.

**Check All Paths:**

Select this option to require that all paths to a trust anchor must be validated. When this option is deselected, all paths to a trust anchor are not validated. By default, the option is deselected.

**Inhibit Policy Mapping:**

Select this option to allow policy mapping in the certification path. When this option is deselect, policy mapping is not allowed in the certification path. By default, the option is deselected.

**LDAP Server:**

Sets the URL or path of the Lightweight Directory Access Protocol (LDAP) server used to retrieve information about the certificate revocation list (CRL). The LDAP server searches for CRL information using the distinguished name (DN) according to the rules specified in RFC 3280

, section 4.2.1.14. For example, you can type `www.ldap.com` for the URL or `ldap://ssl.ldap.com:200` for the path and port. No default value is provided.

**Follow URIs in Certificate AIA:**

Select this option to specify to follow any URIs specified in the certificate's Authority Information Access (AIA) extension for path discovery. The AIA extension specifies where to find up-to-date certificates. When this option is deselected, no URIs are processed in the AIA extension from the certificate. By default, the option is deselected.

**Basic Constraints Extension Required in CA Certificates:**

Select this option to specify that the certificate authority (CA) Basic Constraints certificate extension must be present for CA certificates. Some early German certified root certificates (7 and earlier) are not compliant to RFC3280

and do not contain the basic constraint extension. If it is known that a user's EE certificate chains up to such a German root, deselect this option. When this option is deselected, the presence of the CA Basic Constraints certificate in CA certificates is not required. By default, the value is selected.

**Require Valid Certificate Signature During Chain Building:**

Select this option to require that all Digital Signature Algorithm (DSA) signatures on certificates be valid before a chain is built. For example, in a chain CA > ICA > EE where the signature for EE is not valid, the chain building stops at ICA. EEs are not included in the chain. When this option is deselected, the entire chain is built regardless of whether an invalid DSA signature is encountered. By default, the option is deselected.

*TSP Options Spec*

(Optional) A `TSPOptionSpec` value that represents the settings that define timestamp information applied to the certified signature.

If you provide a literal value, specify the following options.

**Time Stamp Server URL:**

Sets the URL for a TSP server. If no value is provided, the timestamp from the local system is applied. No default value is provided.

**Time Stamp Server Username:**

Sets the user name, if necessary, for accessing the TSP server. No default value is provided.

**Time Stamp Server Password:**

Sets the password for the user name, if necessary, for accessing the TSP server. No default value is provided.

**Time Stamp Server Hash Algorithm:**

Sets the hash algorithm used to digest the request sent to the timestamp provider. Select one of these values:

**SHA1: (Default)**

The Secure Hash Algorithm that has a 160-bit hash value.

**SHA256:**

The Secure Hash Algorithm that has a 256-bit hash value.

**SHA384:**

The Secure Hash Algorithm that has a 384-bit hash value.

**SHA512:**

The Secure Hash Algorithm that has a 512-bit hash value.

**RIPEMD160:**

The RACE Integrity Primitives Evaluation Message Digest that has a 160-bit message digest algorithm and is not FIPS-compliant.

**Revocation Check Style:**

Sets the revocation-checking style used for verifying the trust status of the CRL provider's certificate from its observed revocation status. Select one of these values:

**NoCheck:**

Does not check for revocation.

**BestEffort:**

(Default) Checks for revocation of all certificates when possible.

**CheckIfAvailable:**

Checks for revocation of all certificates only when revocation information is available.

**AlwaysCheck:**

Checks for revocation of all certificates.

**Use Expired Timestamps:**

Select this option to use timestamps that have expired during the validation of the certificate. When this option is deselected, expired timestamps are not used. By default, this option is selected.

**Predicted Time Stamp Token Size (In Bytes):**

Sets the estimated size, in bytes, of the TSP response. The size is used to create a signature hole in the PDF document. This value represents the maximum size of the timestamp response that the configured TSP could return. Configuring an undersized value can cause the operation to fail; however, configuring an oversized value causes the size to be larger than necessary. It is recommended that this value is not modified unless the timestamp server requires a response size to be less than 4096 bytes. Valid values are from `60` to `10240`. The default value is `4096`.

**Send Nonce:**

Select this option to send a nonce with the request. A *nonce* is a parameter that varies with time. These parameters can be a timestamp, a visit counter on a web page, or a special marker. The

parameter is intended to limit or prevent the unauthorized replay or reproduction of a file. When the option deselected, a nonce is not sent with the request. By default, the option is selected.

**Output PDF properties**

Property to specify the output PDF document.

*Output PDF*

The location in the process data model to store the PDF document. The signed PDF document contains the updated validation information. The data type is `document`.

**Exceptions**

This operation can throw PDFOperationException, MissingSignatureFieldException, InvalidArgumentException, SignatureFieldNotSignedException, SignatureVerifyException, and PermissionsException exceptions.

## Add Visible Signature Field operation

Creates a visible signature field in a PDF document. Add a visible signature field when you want the user to see the signature.

For example, your application must be digitally signed by the user but the form has no signature field. You use the Add Visible Signature Field operation to add a signature field to the PDF form. Adding a visible signature field permits a user to see the form when you display it to them. Then, in a subsequent User2.0 operation, the users can sign the PDF form in Workspace.

**IMPORTANT:** You cannot add a visible signature field to a dynamic PDF form.

For information about the General and Route Evaluation property groups, see Commonoperationproperties.

Some properties of this operation provide the following buttons to manage entries in lists:

**Add A List Entry:**

Adds an entry to the list. Depending on the option, you type the information, select an item from a drop-down list, or select a file from a network location or computer. When you select a file from a location on your computer, during run time, the file must exist in the same location on the AEM Forms Server.

**Delete Selected List Entry:**

Removes an entry from the list.

**Move Selected List Entry Up One Row:**

Moves the selected entry up in the list.

**Move Selected List Entry Down One Row:**

Moves the selected entry down in the list.

### Common properties

Properties to specify mandatory values.

*Input PDF*

A `document` value that represents the PDF document to which the invisible signature field is added.

If you provide a literal value, clicking the ellipsis button opens the Select Asset dialog box. (See AboutSelectAsset.)

*Signature Field Name*

A `string` value for the name of the signature field to add. The fully qualified name of the signature field must be specified.

*Page Number*

An `int` value of the page number on which the signature field is added. Valid values are from `1` to the number of pages contained within the document. The maximum allowed value is `9999`. The default value is `0`, which must be changed.

*Position Rectangle*

A `PositionRectangle` value that specifies the position for the signature field is measure in terms of points (pt). This parameter is required and cannot be null. If the specified rectangle is not at least partially on the crop box of the specified page, a PDFOperationException exception is thrown.

If you provide a literal value, specify the following values.

- **Lower Left X:** Sets the lower-left X position of the signature field. The value cannot be zero or a negative number and is relative to the crop box of the page. The default value is `1`.
- **Lower Left Y:** Sets the lower-left X position of the signature field. The value cannot be zero or a negative number and is relative to the crop box of the page. The default value is `1`.
- **Height:** Sets the height of the signature field. The value cannot be zero or a negative number. The default value is `100`.
- **Width:** Sets the width of the signature field. The value cannot be zero or a negative number. The default value is `100`.

### Advanced properties

Properties to specify optional values.

*Field MDP Options Spec*

(Optional) A `FieldMDPOptionSpec` value that specifies the PDF document fields that are locked after the signature field is signed.

If you provide a literal value, you can set the following options.

**Field Locking Action:**

A list that sets the type of action to use to lock fields in a PDF document. Select one of these values:

- **All Fields:** Lock all fields in the PDF document.
- **Include Fields:** Lock only the fields specified in the Application To Form Fields option.
- **Exclude Fields:** Lock all fields except for those specified in the Applicable To Form Fields option.

**Applicable to Form Fields:**

Sets a comma-separated list of fully qualified field names that indicate which fields the action is applicable or not applicable to. This option is available when the Field Locking Action option is set to a value of Include Fields or Exclude Fields.

*Seed Value Options Spec*

(Optional) A `PDFSeedValueOptionSpec` value that represents the seed value dictionary that is associated with a signature field. A seed value dictionary contains entries that constrain information that is used at the time the signature is applied. The options are used for specifying the document signature settings.

If you provide a literal value, you can set the following options:

**Signature Handler Options:**

Options for specifying the filters and subfilters that are used for validating a signature field. The signature field is embedded in a PDF document and the seed value dictionary is associated with a signature field.

- **Signature Handler:** A list of handlers to use for the digital signatures. Adobe.PPKLite is a valid value that can be selected to represent the creation and validation of Adobe-specific signatures. You can use other signature handlers by typing string values, such as `Entrust.PPEF`, `CIC.SignIt`, and `VeriSign.PPKVS`. For information about supported signature handlers, see *PDF Reference.* No default value is selected. The following signature handler is available to be selected from the list:
  - **Adobe.PPKLite:** The recommended handler for signing PDF documents.

    ***Required:*** *Select to specify that the signature handler is used for the seed value. It is not selected by default.*

- **Signature SubFilter:**The supported subfilter names, which describe the encoding of the signature value and key information. Signature handlers must support the listed subfilters; otherwise, the signing fails. These string values, which you must type, are valid for public-key cryptographic (see *PDF Reference*):
  - **adbe.x509.rsa_sha1:**The key contains a DER-encoded PKCS#1 binary data object. The binary objects represent the signature that is obtained as the RSA encryption of the byte range SHA-1 digest with the private key of the signer. Use this value when signing PDF documents using PKCS#1 signatures.
  - **adbe.pkcs7.detached:**The key is a DER-encoded PKCS#7 binary data object that contains the signature. No data is encapsulated in the PKCS#7-signed data field.

  - **adbe.pkcs7.sha1:** The key is a DER-encoded PKCS#7 binary object that represents the signature value. The SHA-1 digest of the byte range digest is encapsulated in the PKCS#7 signed data.

  ***Required:*** *Select to specify that signature subfilters are used for the seed value. It is not selected by default.*

- **Digest Methods:** The list of acceptable hashing algorithms to use. Add an item to the list and select an encryption algorithm. Select one of these values:
  - **SHA1:** (Default) The Secure Hash Algorithm that has a 160-bit hash value.
  - **SHA256:** The Secure Hash Algorithm that has a 256-bit hash value.
  - **SHA384:** The Secure Hash Algorithm that has a 384 bit-hash value.
  - **SHA512:** The Secure Hash Algorithm that has a 512 bit-hash value.
  - **RIPEMD160:**The RACE Integrity Primitives Evaluation Message Digest that has a 160-bit message digest algorithm and is not FIPS-compliant.

  ***Required:*** *Select to specify that the signature encryption algorithms are used for the seed value. It is not selected by default.*

- **Minimum Signature Compatibility Level:** The minimum PDF version to use to sign the signature field. Select one of these values:
  - **PDF 1.5:** Use PDF Version 1.5.
  - **PDF 1.7:** Use PDF Version 1.7.

  ***Required:*** *Select to specify the minimum signature compatibility level is used for the seed value. It is not selected by default.*

**Signature Information:**

A group of options for specifying the reasons, timestamp, and details of the digital signature.

- **Include Revocation Information in Signature:** Select to specify that revocation information must be embedded as part of the signature for long-term validation support. When you deselect this option, the revocation information is not embedded as part of the signature. By default, this option is deselected.

  ***Required:*** *Select to specify that revocation checking is required for the seed value. It is not selected by default.*

- **Signing Reasons:** The list of reasons that are associated with the seed value dictionary used for signing the PDF document. Add an item to the list and type a reason.

  ***Required:*** *Select to specify that the associated reasons are included for the seed value. It is not selected by default.*

- **TimeStamp Server URL:** The URL that specifies the location of the timestamp server to use when signing a PDF document.

  ***Required:*** *Select to specify that the timestamp server is required for the seed value. It is not selected by default.*

- **Signing/Enrollment Server URL:** The location of the server that provides a web service. The web service digitally signs a PDF document or enrolls for new credentials.

***Required:*** *Select to specify that the signing or enrollment server is used for the seed value. It is not selected by default.*

- **Server Type:** The type of server to use for the value specified for the Signing/Enrollment Server URL option. Select one of these values:
  - **Browser:**(Default) The URL references content that is displayed in a web browser to allow enrolling for a new credential if a matching credential is not found.
  - **ASSP:** The URL references a signature web service. The web service is used to digitally sign the PDF document on a server. The server is specified in the Signing/Enrollment Server URL option in this operation.

  ***Required****: Select to use the web service to sign the PDF document. It is not selected by default.*

**Signature Type:**

The changes that are permitted after the signature is added and legal attestations are provided.

- **Type of Signature:** The list representing the type of signatures that can be applied to the signature field. Select one of these values:

  ***Any:*** *(Default) Any type of signature can be applied when filling forms, instantiating page templates, or creating, deleting, and modifying annotations.*

  ***Recipient Signature:*** *Restricts the signer to apply a Four Corner security model on the signature field.*

  ***Certification Signature:*** *Constrains the signer to apply a certification signature on the signature field with specified permissions. The specified permissions are configured in the Field MDP Options Spec property for this operation. Select one of these values:*
  - **No changes allowed**: The end user is not permitted to change the form. Any change invalidates the signature.
  - **Form fill-in and digital signatures**: The end user is permitted to fill the form, instantiate page templates, and sign the form.
  - **Annotations, form fill-in, and digital signatures**: The end user is permitted to fill the form, instantiate page templates, sign the form, and create annotations, deletions and modifications.

- **Legal Attestations:** The list of legal attestations that are associated with the seed value. Legal attestation constraints affect only a certification signature. When you select Any or Certificate Signature option for the Type of Signature, you can add a legal attestation to the list by typing it.

  ***Required:*** *Select to specify that legal attestations are used for the seed value. It is not selected by default.*

**Signing Certificates:**

The list of certificates, keys, issuers, and policies that are used for a digital signature. Add certificates, keys, issuers, and policies to the list by using the Open dialog box.

- **Signing Certificates:** A list of certificates that are used for certifying and verifying a signature.

  ***Required:*** *Select to specify that signing certificates are used for the seed value. It is not selected by default.*

- **Subject Distinguished Name:** The list of dictionaries, where each dictionary contains key value pairs that specify the subject distinguished name (DN). The DN must be present within the certificate for it to be acceptable for signing. Add DNs to the list by using the Add Subject DN dialog box. (See AboutAddSubject DN.)

  ***Required:*** *Select to specify that subject distinguished names are used for the seed value. It is not selected by default.*
- **KeyUsage:** The list of key usage extensions that must be present for signing a certificate. Add an entry to the list and select the key usage. Additional key usage entries are available in *PDF Reference*. Select one of these key usage values for each entry:
  - **Don't Care:**(Default) The key usage extension is optional.
  - **Require Key Usage:** The key usage extension must be present.
  - **Exclude Key Usage:** The key usage extension must not be present.

  ***Required:*** *Select to specify that key usage extensions are used for the seed value. It is not selected by default.*

**Issuers and Policies:**

The list of certificate issuers, policies, and associated object identifiers.

- **Certificate Issuers:** The list of certificate issuers. Add certificate issuers to the list by using the Open dialog box.

  ***Required:*** *Select to specify that certificate issuers are used for the seed value. It is not selected by default.*
- **Certificate Policies And Associated Object Identifiers:** The list of certificate policies that are associated with the certificate seed value. Add certificate policies to the list by typing it.

  ***Required:*** *Select to specify that certificate policies and associated identifies are used for the seed value. It is not selected by default.*

**Output properties**

Property to specify the output PDF document.

*Output PDF*

The location in the process data model to store the PDF document. The PDF document has the additional signature field added to it. The data type is `document`.

**Exceptions**

This operation can throw PDFOperationException, InvalidArgumentException, DuplicateSignatureFieldException, and PermissionsException exceptions.

## Certify PDF operation

Certifies the specified signature field that is located in the PDF document by using the security credential that corresponds to the specified alias. You can apply a certifying signature only if the PDF document

does not already contain other digital signatures. The alias must map to a valid credential that is already located in Trust Store Management. (See Trust Store ManagementHelp

.)

For example, your application must apply certification signature to multiple PDF forms before saving them to network location. The PDF document has a an existing signature field on it. The certification signature must allow a user to sign and fill in the form but not add or remove pages from it. You use the Certify operation to add the certification signature before it is saved to a network location during runtime.

Any combination of encrypting, certifying, and applying usage rights to the same document must occur within a short-lived process and must occur in the following order:

1) Encrypt (Encryption service or the Applypolicy(deprecated) operation of the Rights Management service)
2) Certify PDF
3) ApplyUsageRights (Acrobat Reader DC extensions service)

You can also certify a PDF package. The cover sheet in a certified PDF package can be an interactive form; however, the files contained in the package must not be interactive. The certification for a PDF package becomes compromised if any of the component files are modified, even if the component file's certification allows form fill-in.

To create a certified PDF package, apply operations in the following order. These services must be invoked within a short-lived process:

1) (Optional) For each document to be contained in the PDF package, encrypt, or policy-protect, and then certify. The files in the PDF package cannot be interactive forms; therefore, do not apply usage rights.
2) Assemble the PDF package.
3) (Optional) Encrypt or policy-protect the PDF package.
4) Certify the PDF package.
5) Apply usage rights to the PDF package.

You can also create a process that consumes component files that are already encrypted, policy-protected, or certified. If any of the component files consumed by your process are encrypted or policy-protected, ensure that your DDX file defines them. You design the DDX file so that the Assembler service does not have to open the component files. To prevent opening component files, use the DDX `PackageFiles` source element instead of the DDX `PackageFiles` filter element in the DDX file.

For information about the General and Route Evaluation property groups, see Commonoperationproperties.

### Common properties

Properties to specify the PDF document, credential, and other certification values.

*Input PDF*

A `document` value that represents the PDF document to certify.

If you provide a literal value, clicking the ellipsis button opens the Select Asset dialog box. (See AboutSelectAsset.)

When you provide a PDF document that has signature fields that are signed, it populates the Signature Field Name property as a list. The list contains fully qualified names of the signature fields in the PDF document. If the document is signed, an exception is thrown because a certification signature must be the first signature in the PDF document.

*Certifying Credential*

A `Credential` value that represents the security credential that is used to sign the PDF document.

If you provide a literal value, you can configure the following options.

**Use SPI:**

Select this option to use the credentials from the SPI. When this option is deselected, local credentials are used. By default, the option is deselected.

**Alias:**

Sets an alternative name for a credential managed by the Signature service. By default, a list of all signing credentials is provided. The list of credentials is comprised of Local and HSM credentials configured in Trust Store on the AEM Forms Server.

**SPI Name:**

Sets the name of the SPI that is provided to the Signature service. The SPI is used to extend the digital signatures functionality when credentials are not exposed to the AEM Forms Server. This option is available when the Use SPI option is selected. No default value is provided.

**Certificate:**

Sets the location of the certificate on the file system. No default value is provided. This option is available when the Use SPI option is selected. No default value is provided.

When you click the ellipsis button , the Open dialog box opens. In the dialog box, you can select a file from your computer or from a network location. During run time, if you selected a file from a location on your computer, it must exist in the same location on the AEM Forms Server.

**SPI Properties:**

Sets the location of the properties file to pass custom inputs to an implementation of the SPI. This option is available when the Use SPI option is selected. No default value is provided.

If you provide a literal value, clicking the ellipsis button opens the SPI properties dialog box. (See AboutSPIProperties.) In the dialog box, add, remove, and edit the keys and values for each SPI property. The SPI implementation determines the keys that you provide. For information about creating custom service providers, see Programming withAEM Forms

.

*Signature Field Name*

(Optional) A `string` value that represents the name of the signature field in the PDF document that is signed. Before a PDF document can be signed, the signature field must exist, be unsigned, and have permissions for signing. The fully qualified name of the signature field is specified. If the signature field name is not specified, the Signature service adds an invisible signature field with an automatically generated name.

When using a PDF document based on a form created in Designer, the partial name of the signature field can also be used. For example, `form1[0].#subform[1].SignatureField3[3]` can be specified as `SignatureField3[3]`. If multiple signature fields exist with a similar partial name, the first signature field enumerated with the same partial name is signed. It is recommended that a fully qualified name is used to avoid these situations.

If you provide a literal value for the Signature Field Name property and a literal value is provided in the Input PDF property, a list appears. Select one of the values from the list of fully qualified names. Each fully qualified name represents a signature field in the provided PDF document.

*Document MDP Permissions*

(Optional) An `MDPPermissions` value that represents the permissions that control the actions an end user can perform on a document without making the certification signature invalid.

If you provide a literal value, select one of these values:

- **No Changes Allowed:** No changes to the document are permitted. Any change invalidates the signature.
- **Form Fill-in and Digital Signatures:** Permitted changes include filling in forms, instantiating page templates, and signing.
- **Annotations, Form Fill-in, and Digital Signatures:** Permitted changes include filling in forms and instantiating page templates, as well as creating, deleting, and modifying annotations.

*Digest Hashing Algorithm*

(Optional) A `HashAlgorithm` value that represents the hash algorithm that is used to digest the PDF document.

If a literal value is provided, select one of these values:

**SHA1:**

The Secure Hash Algorithm that has a 160-bit hash value.

**SHA256:**

(Default) The Secure Hash Algorithm that has a 256-bit hash value.

**SHA384:**

The Secure Hash Algorithm that has a 384-bit hash value.

**SHA512:**

The Secure Hash Algorithm that has a 512-bit hash value.

**RIPEMD160:**

The RACE Integrity Primitives Evaluation Message Digest that has a 160-bit message digest algorithm and is not FIPS-compliant.

*Embed Revocation Information*

(Optional) A `boolean` value that specifies whether revocation checking is performed for the signer's certificate. If revocation checking is done, it is embedded in the signature and then used during validation. It also enables the PDF file to be stored for long-term validation. A value of `False` means that revocation-checking is not performed and the values for the OCSP Options Spec property cannot be modified. The default value is `True`, which means that revocation checking is performed and the values for the OCSP Options Spec property can be modified.

If a literal value is provided, by default, the option is selected. When selected, it means that revocation checking is not performed and the values for the OCSP Options Spec property cannot be modified. When the option is deselected, it means that revocation checking is not performed and the values for the OCSP Options Spec property cannot be modified.

*Lock Certifying Signature Field*

(Optional) A `boolean` value that specifies whether the signature field is locked after it is used to certify the document. If the field is locked, the signature field cannot be modified or cleared without the certifying credential. The default setting is `False`, which means the signature field is not locked.

If the Process Documents With Acrobat 9 Compatibility service configuration property is selected, the signature field is locked. (See Signatureserviceconfiguration.)

If a literal value is provided, by default, the option is selected. When selected, it means that the signature field is locked. When the option is deselected, it means that the signature fields are not locked.

**Appearance properties**

Properties to set the appearance of the certification.

*Reason*

(Optional) A `string` value that represents the reason for signing the PDF document.

If you provide a literal value, type a string or choose one of the following values:

- I am the author of this document
- I have reviewed this document
- I am approving this document
- I attest to the accuracy and integrity of this document
- I agree to the terms defined by the placement of my signature on this document
- I agree to the specified portions of this document

*Location*

(Optional) A `string` value that represents the location of the signer.

*Contact Information*

(Optional) A `string` value that represents the contact information, such as an address and a telephone number, of the person who signed the PDF document.

*Legal Attestation*

(Optional) A `string` value that represents an additional explanation of the content that generates warnings. For more information on legal attestations, see the PDFUtilitiesService.

*Appearance Options Spec*

(Optional) A `PDFSignatureAppearanceOptionSpec` value that represents options for the signature appearance. If you provide a literal value, specify the following options.

**Signature Type:**

Sets the appearance type of the signature. The default value is Name. Select one of these values:

- **No Graphic:** The appearance of the signature consists of only the signature text.
- **Graphic:** The appearance of the signature consists of a graphic area and a text area. The graphic area displays the PDF document specified by the Graphic PDF Document option. The text area displays the signature text.
- **Name:** The appearance of the signature consists of a graphic area and a text area. The graphic area displays a graphic of the name of the signer and the text area displays the signature text.

**Graphic PDF Document:**

Sets the graphic that is displayed within the signature if a signature type of Graphic is used. Only a PDF file can be used. This option can be set if Graphic is selected in the Signature Type list.

When you click the ellipsis button , the Open dialog box opens. In the dialog box, you can select a file from your computer or from a network location. During run time, if you selected a file from a location on your computer, it must exist in the same location on the AEM forms Server.

**Use Default Adobe PDF Logo:**

Select this option to display the default Adobe PDF logo within the signature appearance. When this option is deselected, the Adobe PDF logo is not displayed. By default, the option is selected.

**Logo PDF Document:**

Sets a PDF document to display within the signature appearance. The PDF document contains an image to display. This option can be set when the Use Default Adobe PDF Logo option is deselected.

When you click the ellipsis button, the Open dialog box opens. In the dialog box, you can select a file from your computer or from a network location. During run time, if you selected a file from a location on your computer, it must exist in the same location on the AEM forms Server.

**Logo Opacity:**

Sets the opacity of the logo that is displayed within the signature. Valid values are from `0.0` (fully transparent) to `1.0` (fully opaque). Can be set only if Use Default Adobe PDF Logo is deselected. If any value outside this range is specified, the default of `0.50` is used.

**Text Direction:**

Sets the direction of the text displayed within the signature. The default value is Auto. Select one of these values:

- **Auto:** Use the direction specified by the PDF document.
- **Left:** The text direction is left to right.
- **Right:** The text direction is right to left.

**Show Name:**

Select this option to display the name of the signer in the digital signature. When this option is deselected, the name of the signer is not displayed. By default, the option is selected.

**Show Date:**

Select this option to display the date the PDF document was signed in the digital signature. When this option is deselected, the date is not displayed. By default, the option is selected.

**Show Reason:**

Select this option to display the reason the PDF document was signed in the digital signature. When this option is deselected, the reason is not displayed. By default, the option is selected.

**Show Location:**

Select this option to display the location the PDF document was signed in the digital signature. When this option is deselected, the location is not displayed. By default, the option is selected.

**Show Distinguished Name:**

Select this option to display the certificate of the signer in the digital signature. When this option is deselected, the certificate is not displayed. By default, the option is selected.

**Show Labels:**

Select this option to display the labels for the preceding display items. When this option is deselected, the labels for the preceding display items are not displayed. By default, the option is selected.

**Advanced properties**

Properties for setting optional advanced parameters. The OCSP Options Spec and CRL Options Spec properties can be modified when the EmbedRevocationInformation for this operation is set to `True`.

*OCSP Options Spec*

(Optional) An `OCSPOptionSpec` value that represents settings for using Online Certificate Status Protocol (OCSP) revocation checking. To provide a literal value, specify the following options.

**URL to Consult Option:** Sets the list and order of the OCSP servers used to perform the revocation check. Select one of these values:

**UseAIAInCert:**

(Default) Use the URL of an online certificate status protocol server specified in the Authority Information Access (AIA) extension in the certificate. The AIA extension is used to identify how to access certificate authority (CA) information and services for the issuer of the certificate.

**LocalURL:**

Use the specified URL for the OCSP server specified in the OCSP Server URL option.

**UseAIAIfPresentElseLocal:**

Use the URL of the OCSP server specified in the AIA extension in the certificate if present. If the AIA extension is not present in the certificate, use the URL that is configured in the OCSP Server URL.

**UseAIAInSignerCert:**

Use the URL of the OCSP server specified in the AIA extension in the OCSP request of the signer certificate.

**OCSP Server URL:**

Sets the URL of the configured OCSP server. The value is used only when the LocalURL or UseAIAIfPresentElseLocal values are in URL To Consult Option.

**Revocation Check Style:**

Sets the revocation-checking style that is used for verifying the trust status of the CRL provider's certificate from its observed revocation status. Select one of these values:

Sets the URL of the configured OCSP server. The value is used only when the LocalURL or UseAIAIfPresentElseLocal values are in URL To Consult Option.

Revocation Check Style:

Sets the revocation-checking style that is used for verifying the trust status of the CRL provider's certificate from its observed revocation status. Select one of these values:

**NoCheck:**

Does not check for revocation.

**BestEffort:**

Checks for revocation of all certificates when possible.

**CheckIfAvailable:**

(Default) Checks for revocation of all certificates only when revocation information is available.

**AlwaysCheck:**

Checks for revocation of all certificates.

**Max Clock Skew Time (Minutes):**

Sets the maximum allowed skew, in minutes, between response time and local time. Valid skew times are `0` - `2147483647` min. The default value is `5` min.

**Response Freshness Time (Minutes):**

Sets the maximum time, in minutes, for which a preconstructed OCSP response is considered valid. Valid response freshness times are `1`- `2147483647` min. The default value is `525600` min. (one year).

**Send Nonce:**

Select this option to send a nonce with the OCSP request. A *nonce* is a parameter that varies with time. These parameters can be a timestamp, a visit counter on a web page, or a special marker. The parameter is intended to limit or prevent the unauthorized replay or reproduction of a file. When the option deselected, a nonce is not sent with the request. By default, the option is selected.

**Sign OCSP Request:**

Select this option to specify that the OCSP request must be signed. When the option is deselected, the OCSP request does not need be signed. By default, the option deselected.

**Request Signer Credential Alias:**

Sets the credential alias used for signing the OCSP request when signing is enabled.

**Go Online for OCSP:**

Select this option to access the network for OCSP information. The network can be accessed to retrieve OCSP information for OCSP checking. The AEM Forms Server uses embedded and cached OCSP information when possible to reduce the amount of network traffic generated due to OCSP checking. When the option is deselected, OCSP checking is not retrieved from the network, and only embedded and cached OCSP information is used. By default, the option is selected.

**Ignore Validity Dates:**

Select this option to use the OCSP response thisUpdate and nextUpdate times. Ignoring these response times prevents any negative effect on response validity. The thisUpdate and nextUpdate times are retrieved from external sources by using HTTP or LDAP, and can be different for each revocation information. When the option is deselected, the thisUpdate and nextUpdate times are ignored. By default, the option is deselected.

**Allow OCSP NoCheck Extension:**

Select this option to allow an OCSPNoCheck extension in the response signing certificate. An OCSP-NoCheck extension can be present in the OCSP Responder's certificate to prevent infinite loops from occurring during the validation process. When the option is deselected, the OCSPNoCheck extension is not used. By default, the option is selected.

**Require OCSP ISIS-MTT CertHash Extension:**

Select this option to specify that certificate public key hash (CertHash) extensions must be present in OCSP responses. This extension is required for SigQ validation. SigQ compliance requires the CertHash extension to be in the OCSP responder certificate. Select this option when processing for SigQ compliance and supported OCSP responders. When the option is deselected, the CertHash extension presence in the OCSP response is not required. By default, the option is deselected.

Sets the maximum allowed skew, in minutes, between response time and local time. Valid skew times are `0` - `2147483647` min. The default value is `5` min.

Response Freshness Time (Minutes):

Sets the maximum time, in minutes, for which a preconstructed OCSP response is considered valid. Valid response freshness times are `1`- `2147483647` min. The default value is `525600` min. (one year).

Send Nonce:

Select this option to send a nonce with the OCSP request. A *nonce* is a parameter that varies with time. These parameters can be a timestamp, a visit counter on a web page, or a special marker. The parameter is intended to limit or prevent the unauthorized replay or reproduction of a file. When the option deselected, a nonce is not sent with the request. By default, the option is selected.

Sign OCSP Request:

Select this option to specify that the OCSP request must be signed. When the option is deselected, the OCSP request does not need be signed. By default, the option deselected.

Request Signer Credential Alias:

Sets the credential alias used for signing the OCSP request when signing is enabled.

Go Online for OCSP:

Select this option to access the network for OCSP information. The network can be accessed to retrieve OCSP information for OCSP checking. The AEM Forms Server uses embedded and cached OCSP information when possible to reduce the amount of network traffic generated due to OCSP checking. When the option is deselected, OCSP checking is not retrieved from the network, and only embedded and cached OCSP information is used. By default, the option is selected.

Ignore Validity Dates:

Select this option to use the OCSP response thisUpdate and nextUpdate times. Ignoring these response times prevents any negative effect on response validity. The thisUpdate and nextUpdate times are retrieved from external sources by using HTTP or LDAP, and can be different for each

revocation information. When the option is deselected, the thisUpdate and nextUpdate times are ignored. By default, the option is deselected.

Allow OCSP NoCheck Extension:

Select this option to allow an OCSPNoCheck extension in the response signing certificate. An OCSP-NoCheck extension can be present in the OCSP Responder's certificate to prevent infinite loops from occurring during the validation process. When the option is deselected, the OCSPNoCheck extension is not used. By default, the option is selected.

Require OCSP ISIS-MTT CertHash Extension:

Select this option to specify that certificate public key hash (CertHash) extensions must be present in OCSP responses. This extension is required for SigQ validation. SigQ compliance requires the CertHash extension to be in the OCSP responder certificate. Select this option when processing for SigQ compliance and supported OCSP responders. When the option is deselected, the CertHash extension presence in the OCSP response is not required. By default, the option is deselected.

*CRL Options Spec*

(Optional) A `CRLOptionSpec` value that represents the certificate revocation list (CRL) preferences when CRL is used to perform revocation checking. If you provide a literal value, specify the following options.

**Consult Local URI First:**

Select this option to use the CRL location provided as a local URI before any specified locations within a certificate. The CRL location provided is used for revocation checking. When this option is selected, it means the local URI is used first. When this option is deselected, the locations specified in the certificate before using the local URI are used. By default, the option is deselected.

**Local URI for CRL Lookup:**

Sets the URL for the local CRL store. This value is used only if the Consult Local URI First option is selected. No default value is provided.

**Revocation Check Style:**

Sets the revocation-checking style used for verifying the trust status of the CRL provider's certificate from its observed revocation status. Select one of these values:

- **NoCheck:** Does not check for revocation.
- **BestEffort:** (Default) Checks for revocation of all certificates when possible.
- **CheckIfAvailable:** Checks for revocation of all certificates only when revocation information is available.
- **AlwaysCheck:** Checks for revocation of all certificates.

**LDAP Server:**

Sets the URL or path of the Lightweight Directory Access Protocol (LDAP) server used to retrieve information about the certificate revocation list (CRL). The LDAP server searches for CRL information using the distinguished name (DN) according to the rules specified in RFC 3280

, section 4.2.1.14. For example, you can type `www.ldap.com` for the URL or `ldap://ssl.ldap.com:200` for the path and port. No default value is provided.

**Go Online for CRL Retrieval:**

Select this option to access the network to retrieve CRL information. CRL information is cached on the server to improve network performance. CRL information is retrieved online only when necessary. When this option is deselected, CRL information is not retrieved online. By default, the option is selected.

**Ignore Validity Dates:**

Select this option to use thisUpdate and nextUpdate times. Ignoring the response's thisUpdate and nextUpdate times prevents any negative effect on response validity. The thisUpdate and nextUpdate times are retrieved from external sources by using HTTP or LDAP and can be different for each revocation information. When the option is deselected, the thisUpdate and nextUpdate time are ignored. By default, the option deselected.

**Require AKI Extension in CRL:**

Select this option to specify that the Authority Key Identifier (AKI) extension must be present in the CRL. The AKI extension can be used for CRL validation. When this option is deselected, the presence of the AKI extension the CRL is not required. By default, the option is deselected.

*TSP Options Spec*

(Optional) A `TSPOptionSpec` value that represents the settings that define timestamp information applied to the certified signature.

If you provide a literal value, specify the following options.

**Time Stamp Server URL:**

Sets the URL for a TSP server. If no value is provided, the timestamp from the local system is applied. No default value is provided.

**Time Stamp Server Username:**

Sets the user name, if necessary, for accessing the TSP server. No default value is provided.

**Time Stamp Server Password:**

Sets the password for the user name, if necessary, for accessing the TSP server. No default value is provided.

**Time Stamp Server Hash Algorithm:**

Sets the hash algorithm used to digest the request sent to the timestamp provider. Select one of these values:

**SHA1: (Default)**

The Secure Hash Algorithm that has a 160-bit hash value.

**SHA256:**

The Secure Hash Algorithm that has a 256-bit hash value.

**SHA384:**

The Secure Hash Algorithm that has a 384-bit hash value.

**SHA512:**

The Secure Hash Algorithm that has a 512-bit hash value.

**RIPEMD160:**

The RACE Integrity Primitives Evaluation Message Digest that has a 160-bit message digest algorithm and is not FIPS-compliant.

**Predicted Time Stamp Token Size (In Bytes):**

Sets the estimated size, in bytes, of the TSP response. The size is used to create a signature hole in the PDF document. This value represents the maximum size of the timestamp response that the configured TSP could return. Configuring an undersized value can cause the operation to fail; however, configuring an oversized value causes the size to be larger than necessary. It is recommended that this value is not modified unless the timestamp server requires a response size to be less than 4096 bytes. Valid values are from `60` to `10240`. The default value is `4096`.

**Send Nonce:**

Select this option to send a nonce with the request. A *nonce* is a parameter that varies with time. These parameters can be a timestamp, a visit counter on a web page, or a special marker. The parameter is intended to limit or prevent the unauthorized replay or reproduction of a file. When the option deselected, a nonce is not sent with the request. By default, the option is selected.

**Output properties**

Property to specify the certified PDF document.

*Output PDF*

The location in the process data model to store the certified PDF document. The data type is `document`. This PDF document is new and the input PDF document is not modified.

### Exceptions

This operation can throw PDFOperationException, SigningException, PermissionsException, InvalidArgumentException, SeedValueValidationException, MissingSignatureFieldException, MissingSignatureFieldException, CredentialLoginException, and FIPSComplianceExceptionexceptions.

## Clear Signature Field operation

Removes the signature from a signature field.

For example, your application must remove digital signatures from PDF documents of people that left the organization. You use the Clear Signature Field operation to remove the digital signature from a specific signature field on the PDF document. In a step before the Clear Signature Field operation, use the GetSignature or GetSignatureField List to determine the name of the signature fields in the PDF document.

For information about the General and Route Evaluation property groups, see Commonoperationproperties.

### Input properties

Properties to specify the PDF document and the signature field.

*Input PDF*

A `document` value that represents a PDF document that contains the signature field to clear.

If you provide a literal value, clicking the ellipsis button opens the Select Asset dialog box. (See AboutSelectAsset.)

When you provide a PDF document that has signature fields that are signed, it populates the Signature Field Name property as a list. The list contains fully qualified names of the signed signature fields in the PDF document.

*Signature Field Name*

A `string` value that represents the name of the signature field to clear. The fully qualified name of the signature field must be specified. When using a PDF document based on a form created in Designer, the partial name of the signature field can be used. For example, `form1[0].#subform[1].SignatureField3[3]` can be specified as `SignatureField3[3]`.

If multiple signature fields exist with a similar partial name, the first signature field enumerated with the same partial name is signed. It is recommended that a fully qualified name is used to avoid these situations.

If you provide a literal value for the Signature Field Name property and a literal value is provided in the Input PDF property, a list appears. Select one of the values from the list of fully qualified names. Each fully qualified name represents a signed signature field in the provided PDF document.

### Output properties

Property to specify the output PDF document.

*Output PDF*

The location in the process data model to store the PDF document. The PDF document contains the cleared signature field. The data type is `document`.

### Exceptions

This operation can throw PDFOperationException, PermissionsException, InvalidArgumentException, and MissingSignatureFieldException exceptions.

## Get Certifying Signature Field operation

Returns the signature field that was used to certify the PDF document and shell PDF forms.

For example, your application must retrieve the certifying signature from a PDF document and validate it. The PDF document may have more than one digital signature on it. You use the Get Certifying Signature Field operation to retrieve the signature field that contains the certifying signature.

For information about the General and Route Evaluation property groups, see Commonoperationproperties.

### Input properties

Property to specify a certified PDF document.

*Input PDF*

A `document` value that represents a PDF document that contains a signature field that contains a certified signature.

If you provide a literal value, clicking the ellipsis button opens the Select Asset dialog box. (See AboutSelectAsset.)

### Output properties

Property to specify a signature field.

*Certifying Signature Field Result*

The location in the process data model to store the information about the certified signature field. The data type is `PDFSignatureField`.

### Exceptions

This operation can throw PDFOperationException, InvalidArgumentException, NoCertifyingSignatureException, and PermissionsException exceptions.

## Get Legal Attestation operation

Retrieves the legal attestation values for the document. Legal attestation is attributed to a certification signature. There is only one certification signature present in a PDF document. For more information on legal attestations, see the PDFUtilitiesService.

For example, your application must retrieve attestations for digital signatures in a PDF document. You use the Get Legal Attestation operation to retrieve legal warnings that pertain to the certification signature in the PDF document.

For information about the General and Route Evaluation property groups, see Commonoperationproperties.

### Input properties

Properties to specify the input PDF document and whether to generate legal attestations.

*Input PDF*

A `document` value that represents a PDF document.

If you provide a literal value, clicking the ellipsis button opens the Select Asset dialog box. (See AboutSelectAsset.)

*Generate Attestations On The Fly*

(Optional) A `boolean` value that determines whether the legal attestation counter values are calculated or picked up from the dictionary. The default value is `False`, which means that the counter values are picked up from the pre-existing dictionary.

### Output properties

Property to specify the legal warnings result.

*Legal Warnings Result*

The location in the process data model to store information about legal warnings in the document. The data type is `PDFLegalWarnings`.

### Exceptions

This operation can throw PDFOperationException, InvalidArgumentException, NoCertifyingSignatureException, and PermissionsException exceptions.

## Get Signature operation

Returns signature data for a given signature field.

For example, your application must get the signature information from a signed PDF document. You require the timestamp and subfilter information to validate the signature with an external web service.

You use the Get Signature operation to retrieve the timestamp and subfilter information from the digital signature.

For information about the General and Route Evaluation property groups, see Commonoperationproperties.

### Input properties

Properties to specify the PDF document and signature field.

*Input PDF*

A `document` value that contains a signature field for which data is retrieved.

If you provide a literal value, clicking the ellipsis button opens the Select Asset dialog box. (See AboutSelectAsset.)

When you provide a PDF document that has signature fields that are signed, it populates the Signature Field Name property as a list. The list contains fully qualified names of the signed signature fields in the PDF document.

*Signature Field Name*

A `string` value that represents the name of the signature field that contains a signature. The fully qualified name of the signature field must be specified. When using a PDF document based on a form created in Designer, the partial name of the signature field can be used. For example, `form1[0].#subform[1].SignatureField3[3]` can be specified as `SignatureField3[3]`.

If you are getting signature fields from a PDF document, the partial name of the signature field can also be used. For example, `form1[0].#subform[1].SignatureField3[3]` can be specified as `SignatureField3[3]`. If multiple signature fields exist with a similar partial name, the first signature field enumerated with the same partial name is signed. It is recommended that a fully qualified name is used to avoid these situations.

If you provide a literal value for the Signature Field Name property and a literal value is provided in the Input PDF property, a list appears. Select one of the values from the list of fully qualified names. Each fully qualified name represents a signed signature field in the provided PDF document.

### Output properties

Property to specify the signature data.

*PDF Signature Result*

The location in the process data model to store the value that contains the signature and information about the filter and subfilters. The filters and subfilters are used for validating the signature. The value also contains information about certificates that are embedded in the PDF document. The data type is a `PDFSignature` value.

### Exceptions

This operation can throw PDFOperationException, SignatureFieldNotSignedException, InvalidArgument-Exception, MissingSignatureFieldException, and PermissionsException exceptions.

## Get Signature Field List operation

Returns a list of all the signature fields located within a PDF document. You can retrieve the names of all the signature fields in a PDF document for signing or certifying.

For example, your application must use the AEM Forms Server to sign signature fields on the PDF form, however, you do not know the names of the signature fields. You use the Get Signature Field List operation to retrieve the list of signature fields from the PDF form. You get the name of the signature field to sign from the list.

For information about the General and Route Evaluation property groups, see Commonoperationproperties.

### Input properties

Property to specify the PDF document.

*Input PDF*

A `document` value that contains a signature field for which data is retrieved.

If you provide a literal value, clicking the ellipsis button opens the Select Asset dialog box. (See AboutSelectAsset.)

### Output properties

Property to specify the signature field list.

*Signature Field List*

The location in the process data model to store signature fields in the PDF document. The data type is a `list` of `PDFSignatureField` values.

For information about retrieving values from a list, see Accessingdatain data collections.

### Exceptions

This operation can throw PDFOperationException, InvalidArgumentException, and PermissionsException exceptions.

## Get Signed Version operation

Returns the version of the PDF document that corresponds to the document state when the specified signature field was signed or certified.

For example, your application must retrieve the version of the PDF document at the time that it was signed. You use the Get Signed Version operation to retrieve the version of the PDF document when it was signed. It is possible to modify unlocked fields, or incremental updates to a signed PDF form. You get the signed version because to see the values of fields in the PDF document at the time it was signed.

For information about the General and Route Evaluation property groups, see Commonoperationproperties.

**Input properties**

Properties to specify the PDF document and signature field name.

*Input PDF*

A `document` value that represents a signed PDF document.

If you provide a literal value, clicking the ellipsis button opens the Select Asset dialog box. (See AboutSelectAsset.)

When you provide a PDF document that has signature fields that are signed, it populates the Signature Field Name property as a list. The list contains fully qualified names of the signed signature fields in the PDF document.

*Signature Field Name*

A `string` value that represents the name of the signature field in the PDF document that contains a signature. The fully qualified name of the signature field must be specified. When using a PDF document based on a form created in Designer, the partial name of the signature field can be used. For example, `form1[0].#subform[1].SignatureField3[3]` can be specified as `SignatureField3[3]`.

If you are retrieving signature fields from a PDF document, the partial name of the signature field can also be used. For example, `form1[0].#subform[1].SignatureField3[3]` can be specified as `SignatureField3[3]`. If multiple signature fields exist with a similar partial name, the first signature field enumerated with the same partial name is signed. It is recommended that a fully qualified name is used to avoid these situations.

If you provide a literal value for Signature Name Field and a literal value is provided in the Input PDF property, a list appears. Select one of the values from the list of fully qualified names. Each fully qualified name represents a signed signature field in the provided PDF document.

**Output properties**

Property to specify the PDF document.

*Output PDF*

The location in the process data model to store the signed PDF document. The data type is `document`.

**Exceptions**

This operation can throw PDFOperationException, SigningException, SignatureFieldNotSignedException, InvalidArgumentException, MissingSignatureFieldException, and PermissionsException exceptions.

## Modify Signature Field operation

Modifies the field lock and seed value options and field MDP options of an unsigned signature field in a PDF document. A field lock value specifies a list of fields that are locked when a signature field is signed. A locked field prevents users from making changes to the field. A seed value contains constraining information that is used at the time the signature is applied, such as the actions that can occur without invalidating the signature.

For example, your application must lock all the fields after the PDF document is signed. The existing signature field might lock only one field after a digital signature is applied. You use the Modify Signature Field operation to change the signature field so that all fields are locked after a signature is applied.

For information about the General and Route Evaluation property groups, see Commonoperationproperties.

Some properties of this operation provide the following buttons to manage entries in lists:

**Add A List Entry:**

Adds an entry to the list. Depending on the option, you type the information, select an item from a drop-down list, or select a file from a network location or computer. When you select a file from a location on your computer, during run time, the file must exist in the same location on the AEM Forms Server.

**Delete Selected List Entry:**

Removes an entry from the list.

**Move Selected List Entry Up One Row:**

Moves the selected entry up in the list.

**Move Selected List Entry Down One Row:**

Moves the selected entry down in the list.

### Input properties

Properties to specify the PDF document with an unsigned signature field, the name of the signature field, and the signature field properties.

*Input PDF*

A `document` value that represents the PDF document where the signature field is modified.

If you provide a literal value, clicking the ellipsis button opens the Select Asset dialog box. (See AboutSelectAsset.)

When you provide a PDF document that has unsigned signature fields, it populates the Signature Field Name property as a list. The list contains fully qualified names of unsigned signature fields in the PDF document.

*Signature Field Name*

A `string` value that represents the name of the signature field in the PDF document that contains a signature. The fully qualified name of the signature field must be specified. When using a PDF document based on a form created in Designer, the partial name of the signature field can be used. For example, `form1[0].#subform[1].SignatureField3[3]` can be specified as `SignatureField3[3]`.

If you are modifying signature fields in a PDF document, the partial name of the signature field can also be used. For example, `form1[0].#subform[1].SignatureField3[3]` can be specified as `SignatureField3[3]`. If multiple signature fields exist with a similar partial name, the first signature field enumerated with the same partial name is signed. It is recommended that a fully qualified name is used to avoid these situations.

If you provide a literal value for Signature Name Field and a literal value is provided in the Input PDF property, a list appears. Select one of the values from the list of fully qualified names. Each fully qualified name represents an unsigned signature field in the provided PDF document.

*Field MDP Options Spec*

(Optional) A `FieldMDPOptionSpec` value that specifies the PDF document fields that are locked after the signature field is signed.

If you provide a literal value, you can set the following options.

**Field Locking Action:**

A list that sets the type of action to use to lock fields in a PDF document. Select one of these values:

- **All Fields:** Lock all fields in the PDF document.
- **Include Fields:** Lock only the fields specified in the Application To Form Fields option.
- **Exclude Fields:** Lock all fields except for those specified in the Applicable To Form Fields option.

**Applicable to Form Fields:**

Sets a comma-separated list of fully qualified field names that indicate which fields the action is applicable or not applicable to. This option is available when the Field Locking Action option is set to a value of Include Fields or Exclude Fields.

*Seed Value Options Spec*

(Optional) A `PDFSeedValueOptionSpec` value that represents the seed value dictionary that is associated with a signature field. A seed value dictionary contains entries that constrain information that is used at the time the signature is applied. The options are used for specifying the document signature settings.

If you provide a literal value, you can set the following options:

**Signature Handler Options:**

Options for specifying the filters and subfilters that are used for validating a signature field. The signature field is embedded in a PDF document and the seed value dictionary is associated with a signature field.

- **Signature Handler:** A list of handlers to use for the digital signatures. Adobe.PPKLite is a valid value that can be selected to represent the creation and validation of Adobe-specific signatures. You can use other signature handlers by typing string values, such as `Entrust.PPEF`, `CIC.SignIt`, and `VeriSign.PPKVS`. For information about supported signature handlers, see *PDFUtilitiesService.* No default value is selected. The following signature handler is available to be selected from the list:
  - **Adobe.PPKLite:** The recommended handler for signing PDF documents.

    ***Required:*** *Select to specify that the signature handler is used for the seed value. It is not selected by default.*

- **Signature SubFilter:**The supported subfilter names, which describe the encoding of the signature value and key information. Signature handlers must support the listed subfilters; otherwise, the signing fails. These string values, which you must type, are valid for public-key cryptographic (see *PDF Utilities Service*):
  - **adbe.x509.rsa_sha1:**The key contains a DER-encoded PKCS#1 binary data object. The binary objects represent the signature that is obtained as the RSA encryption of the byte range SHA-1 digest with the private key of the signer. Use this value when signing PDF documents using PKCS#1 signatures.
  - **adbe.pkcs7.detached:**The key is a DER-encoded PKCS#7 binary data object that contains the signature. No data is encapsulated in the PKCS#7-signed data field.
  - **adbe.pkcs7.sha1:** The key is a DER-encoded PKCS#7 binary object that represents the signature value. The SHA-1 digest of the byte range digest is encapsulated in the PKCS#7 signed data.

  ***Required:*** *Select to specify that signature subfilters are used for the seed value. It is not selected by default.*

- **Digest Methods:** The list of acceptable hashing algorithms to use. Add an item to the list and select an encryption algorithm. Select one of these values:
  - **SHA1:** (Default) The Secure Hash Algorithm that has a 160-bit hash value.
  - **SHA256:** The Secure Hash Algorithm that has a 256-bit hash value.
  - **SHA384:** The Secure Hash Algorithm that has a 384 bit-hash value.
  - **SHA512:** The Secure Hash Algorithm that has a 512 bit-hash value.
  - **RIPEMD160:**The RACE Integrity Primitives Evaluation Message Digest that has a 160-bit message digest algorithm and is not FIPS-compliant.

  ***Required:*** *Select to specify that the signature encryption algorithms are used for the seed value. It is not selected by default.*

- **Minimum Signature Compatibility Level:** The minimum PDF version to use to sign the signature field. Select one of these values:
  - **PDF 1.5:** Use PDF Version 1.5.
  - **PDF 1.7:** Use PDF Version 1.7.

  ***Required:*** *Select to specify the minimum signature compatibility level is used for the seed value. It is not selected by default.*

**Signature Information:**

A group of options for specifying the reasons, timestamp, and details of the digital signature.

- **Include Revocation Information in Signature:** Select to specify that revocation information must be embedded as part of the signature for long-term validation support. When you deselect this option, the revocation information is not embedded as part of the signature. By default, this option is deselected.

  ***Required:*** *Select to specify that revocation checking is required for the seed value. It is not selected by default.*
- **Signing Reasons:** The list of reasons that are associated with the seed value dictionary used for signing the PDF document. Add an item to the list and type a reason.

  ***Required:*** *Select to specify that the associated reasons are included for the seed value. It is not selected by default.*
- **TimeStamp Server URL:** The URL that specifies the location of the timestamp server to use when signing a PDF document.

  ***Required:*** *Select to specify that the timestamp server is required for the seed value. It is not selected by default.*
- **Signing/Enrollment Server URL:** The location of the server that provides a web service. The web service digitally signs a PDF document or enrolls for new credentials.

  ***Required:*** *Select to specify that the signing or enrollment server is used for the seed value. It is not selected by default.*
- **Server Type:** The type of server to use for the value specified for the Signing/Enrollment Server URL option. Select one of these values:
  - **Browser:**(Default) The URL references content that is displayed in a web browser to allow enrolling for a new credential if a matching credential is not found.
  - **ASSP:** The URL references a signature web service. The web service is used to digitally sign the PDF document on a server. The server is specified in the Signing/Enrollment Server URL option in this operation.

  ***Required****: Select to use the web service to sign the PDF document. It is not selected by default.*

**Signature Type:**

The changes that are permitted after the signature is added and legal attestations are provided.

- **Type of Signature:** The list representing the type of signatures that can be applied to the signature field. Select one of these values:

  ***Any:*** *(Default) Any type of signature can be applied when filling forms, instantiating page templates, or creating, deleting, and modifying annotations.*

  ***Recipient Signature:*** *Restricts the signer to apply a Four Corner security model on the signature field.*

  ***Certification Signature:*** *Constrains the signer to apply a certification signature on the signature field with specified permissions. The specified permissions are configured in the Field MDP Options Spec property for this operation. Select one of these values:*

- **No changes allowed**: The end user is not permitted to change the form. Any change invalidates the signature.
- **Form fill-in and digital signatures**: The end user is permitted to fill the form, instantiate page templates, and sign the form.
- **Annotations, form fill-in, and digital signatures**: The end user is permitted to fill the form, instantiate page templates, sign the form, and create annotations, deletions and modifications.

- **Legal Attestations:** The list of legal attestations that are associated with the seed value. Legal attestation constraints affect only a certification signature. When you select Any or Certificate Signature option for the Type of Signature, you can add a legal attestation to the list by typing it.

  ***Required:*** *Select to specify that legal attestations are used for the seed value. It is not selected by default.*

**Signing Certificates:**

The list of certificates, keys, issuers, and policies that are used for a digital signature. Add certificates, keys, issuers, and policies to the list by using the Open dialog box.

- **Signing Certificates:** A list of certificates that are used for certifying and verifying a signature.

  ***Required:*** *Select to specify that signing certificates are used for the seed value. It is not selected by default.*
- **Subject Distinguished Name:** The list of dictionaries, where each dictionary contains key value pairs that specify the subject distinguished name (DN). The DN must be present within the certificate for it to be acceptable for signing. Add DNs to the list by using the Add Subject DN dialog box. (See AboutAddSubject DN.)

  ***Required:*** *Select to specify that subject distinguished names are used for the seed value. It is not selected by default.*
- **KeyUsage:** The list of key usage extensions that must be present for signing a certificate. Add an entry to the list and select the key usage. Additional key usage entries are available in *PDF Utilities Service*. Select one of these key usage values for each entry:
  - **Don't Care:**(Default) The key usage extension is optional.
  - **Require Key Usage:** The key usage extension must be present.
  - **Exclude Key Usage:** The key usage extension must not be present.

  ***Required:*** *Select to specify that key usage extensions are used for the seed value. It is not selected by default.*

**Issuers and Policies:**

The list of certificate issuers, policies, and associated object identifiers.

- **Certificate Issuers:** The list of certificate issuers. Add certificate issuers to the list by using the Open dialog box.

  ***Required:*** *Select to specify that certificate issuers are used for the seed value. It is not selected by default.*

- **Certificate Policies And Associated Object Identifiers:** The list of certificate policies that are associated with the certificate seed value. Add certificate policies to the list by typing it.

  ***Required:*** *Select to specify that certificate policies and associated identifies are used for the seed value. It is not selected by default.*

**Outputproperties**

Property to specify the PDF document.

*Output PDF*

The location in the process data model to store the modified PDF document. The data type is `document`.

**Exceptions**

This operation can throw PDFOperationException, SigningException, SignatureFieldNotSignedException, InvalidArgumentException, MissingSignatureFieldException, and PermissionsException exceptions.

## Remove Signature Field operation

Removes the specified signature field. You cannot remove a signature field if it is signed or certified. To remove a signature, use the Clear Signature Field operation. (See ClearSignatureField.)

For example, your application must remove all unsigned signature fields a PDF document before it is archived. You use the Remove Signature Field operation to remove unsigned signature fields from the PDF document.

For information about the General and Route Evaluation property groups, see Commonoperationproperties.

**Input properties**

Properties to specify the PDF document with a signed signature field and the name of the signature field.

*Input PDF*

A `document` value that represents a PDF document that contains a signature field to remove.

If you provide a literal value, clicking the ellipsis button opens the Select Asset dialog box. (See AboutSelectAsset.)

When you provide a PDF document that has unsigned signature fields, it populates the Signature Field Name property as a list. The list contains fully qualified names of the unsigned signature fields in the PDF document.

*Signature Field Name*

A `string` value that represents the name of the signature field to remove. The fully qualified name of the signature field can be specified. When using a PDF document based on a form created in Designer, the partial name of the signature field can be used. For example, `form1[0].#subform[1].SignatureField3[3]` can be specified as `SignatureField3[3]`.

If you are removing a signature field from a PDF document, the partial name of the signature field can also be used. For example, `form1[0].#subform[1].SignatureField3[3]` can be specified as `SignatureField3[3]`. If multiple signature fields exist with a similar partial name, the first signature field enumerated with the same partial name is signed. It is recommended that a fully qualified name is used to avoid these situations.

If you provide a literal value for the Signature Field Name property and a literal value is provided in the Input PDF property, a list appears. Select one of the values from the list of fully qualified names. Each fully qualified name represents an unsigned signature field in the provided PDF document.

**Output properties**

Property to specify the output PDF document that has its signature field removed.

*Output PDF*

The location in the process data model to store the PDF document that has the signature field removed. The data type is `document`.

**Exceptions**

This operation can throw PDFOperationException, PermissionsException, InvalidArgumentException, MissingSignatureFieldException, and SignatureFieldSignedException exceptions.

## Sign Signature Field operation

Signs the specified signature field that is located within the PDF document with the security credential that corresponds to the specified alias. Before a PDF document can be signed, the signature field must exist, be unsigned, and have permissions for signing. The alias maps to a credential that is located within the Signature service.

For example, your application must have a person click Complete in Workspace when they approve a PDF form. After clicking Complete, the signature field on the PDF form must be signed by the AEM Forms Server with a corporate certificate. Signing a signature field automatically removes the need for each user to sign the form using Acrobat or Adobe Reader. You use the Sign Signature Field operation to sign the signature field on the server.

For information about the General and Route Evaluation property groups, see Commonoperationproperties.

**Commonproperties**

Properties to specify the input PDF document and its signature attributes.

*Input PDF*

A `document` value that represents the PDF document to sign.

If you provide a literal value, clicking the ellipsis button opens the Select Asset dialog box. (See AboutSelectAsset.)

When you provide a PDF document that has unsigned signature fields, it populates the Signature Field Name property as a list. The list contains fully qualified names of the unsigned signature fields in the PDF document.

*Signing Credential*

A `Credential` value that represents the security credential that is used to sign the PDF document.

If you provide a literal value, you can configure the following options.

**Use SPI:**

Select this option to use the credentials from the SPI. When this option is deselected, local credentials are used. By default, the option is deselected.

**Alias:**

Sets an alternative name for a credential managed by the Signature service. By default, a list of all signing credentials is provided. The list of credentials is comprised of Local and HSM credentials configured in Trust Store on the AEM Forms Server.

**SPI Name:**

Sets the name of the SPI that is provided to the Signature service. The SPI is used to extend the digital signatures functionality when credentials are not exposed to the AEM Forms Server. This option is available when the Use SPI option is selected. No default value is provided.

**Certificate:**

Sets the location of the certificate on the file system. No default value is provided. This option is available when the Use SPI option is selected. No default value is provided.

When you click the ellipsis button , the Open dialog box opens. In the dialog box, you can select a file from your computer or from a network location. During run time, if you selected a file from a location on your computer, it must exist in the same location on the AEM Forms Server.

**SPI Properties:**

Sets the location of the properties file to pass custom inputs to an implementation of the SPI. This option is available when the Use SPI option is selected. No default value is provided.

If you provide a literal value, clicking the ellipsis button  opens the SPI properties dialog box. (See AboutSPIProperties.) In the dialog box, add, remove, and edit the keys and values for each SPI property. The SPI implementation determines the keys that you provide. For information about creating custom service providers, see Programming withAEM Forms

.

*Signature Field Name*

A `string` value that represents the name of the signature field in the PDF document that is signed. The fully qualified name of the signature field can be specified. If you are signing a PDF document, the partial

name of the signature field can also be used. For example, `form1[0].#subform[1].SignatureField3[3]` can be specified as `SignatureField3[3]`.

If multiple signature fields exist with a similar partial name, the first signature field enumerated with the same partial name is signed. If the partial name is already signed, a `DSSSignatureFieldSignedException` is thrown. It is recommended that a fully qualified name is used to avoid these situations.

If the signature field name is not specified, the Signature service adds an invisible signature field with an automatically generated name.

If you provide a literal value for the Signature Field Name property and a literal value is provided in the Input PDF property, a list appears. Select one of the values from the list of fully qualified names. Each fully qualified name represents an unsigned signature field in the provided PDF document.

*Digest Hashing Algorithm*

(Optional) A `HashAlgorithm` value that represents the hash algorithm that is used to digest the PDF document.

If a literal value is provided, select one of these values:

**SHA1:**

The Secure Hash Algorithm that has a 160-bit hash value.

**SHA256:**

(Default) The Secure Hash Algorithm that has a 256-bit hash value.

**SHA384:**

The Secure Hash Algorithm that has a 384-bit hash value.

**SHA512:**

The Secure Hash Algorithm that has a 512-bit hash value.

**RIPEMD160:**

The RACE Integrity Primitives Evaluation Message Digest that has a 160-bit message digest algorithm and is not FIPS-compliant.

*Embed Revocation Information*

(Optional) A `boolean` value that specifies whether revocation checking is performed for the signer's certificate. If revocation checking is done, it is embedded in the signature. The default setting is `False`, which means that revocation checking is not performed.

**Appearance properties**

Properties to set the appearance of the certification.

*Reason*

(Optional) A `string` value that represents the reason for signing the PDF document. If you provide a literal value, type a value or choose one of the following values:

```
I am the author of this document
I have reviewed this document
I am approving this document
I attest to the accuracy and integrity of this document
I agree to the terms defined by the placement of my signature on this
document
I agree to the specified portions of this document
```

*Location*

(Optional) A `string` value that represents the location of the signer.

*Contact Information*

(Optional) A `string` value that represents the contact information, such as an address and a telephone number, of the person who signed the PDF document.

*Appearance Options Spec*

(Optional) A `PDFSignatureAppearanceOptionSpec` value that represents options for the signature appearance. If you provide a literal value, specify the following options.

**Signature Type:**

Sets the appearance type of the signature. The default value is Name. Select one of these values:

- **No Graphic:** The appearance of the signature consists of only the signature text.
- **Graphic:** The appearance of the signature consists of a graphic area and a text area. The graphic area displays the PDF document specified by the Graphic PDF Document option. The text area displays the signature text.
- **Name:** The appearance of the signature consists of a graphic area and a text area. The graphic area displays a graphic of the name of the signer and the text area displays the signature text.

**Signing Format:**

Sets the signing format of the signature. This format is used while signing the form. It is a mandatory. You can select PKCS7Detached or CAdES. The default value is PKCS7Detached.

**Graphic PDF Document:**

Sets the graphic that is displayed within the signature if a signature type of Graphic is used. Only a PDF file can be used. This option can be set if Graphic is selected in the Signature Type list.

When you click the ellipsis button , the Open dialog box opens. In the dialog box, you can select a file from your computer or from a network location. During run time, if you selected a file from a location on your computer, it must exist in the same location on the AEM forms Server.

**Use Default Adobe PDF Logo:**

Select this option to display the default Adobe PDF logo within the signature appearance. When this option is deselected, the Adobe PDF logo is not displayed. By default, the option is selected.

**Logo PDF Document:**

Sets a PDF document to display within the signature appearance. The PDF document contains an image to display. This option can be set when the Use Default Adobe PDF Logo option is deselected.

When you click the ellipsis button, the Open dialog box opens. In the dialog box, you can select a file from your computer or from a network location. During run time, if you selected a file from a location on your computer, it must exist in the same location on the AEM forms Server.

**Logo Opacity:**

Sets the opacity of the logo that is displayed within the signature. Valid values are from `0.0` (fully transparent) to `1.0` (fully opaque). Can be set only if Use Default Adobe PDF Logo is deselected. If any value outside this range is specified, the default of `0.50` is used.

**Text Direction:**

Sets the direction of the text displayed within the signature. The default value is Auto. Select one of these values:

- **Auto:** Use the direction specified by the PDF document.
- **Left:** The text direction is left to right.
- **Right:** The text direction is right to left.

**Show Name:**

Select this option to display the name of the signer in the digital signature. When this option is deselected, the name of the signer is not displayed. By default, the option is selected.

**Show Date:**

Select this option to display the date the PDF document was signed in the digital signature. When this option is deselected, the date is not displayed. By default, the option is selected.

**Show Reason:**

Select this option to display the reason the PDF document was signed in the digital signature. When this option is deselected, the reason is not displayed. By default, the option is selected.

**Show Location:**

Select this option to display the location the PDF document was signed in the digital signature. When this option is deselected, the location is not displayed. By default, the option is selected.

**Show Distinguished Name:**

Select this option to display the certificate of the signer in the digital signature. When this option is deselected, the certificate is not displayed. By default, the option is selected.

**Show Labels:**

Select this option to display the labels for the preceding display items. When this option is deselected, the labels for the preceding display items are not displayed. By default, the option is selected.

**Advanced properties**

Properties for setting optional advanced properties.

*OCSP Options Spec*

(Optional) An `OCSPOptionSpec` value that represents settings for using Online Certificate Status Protocol (OCSP) revocation checking. To provide a literal value, specify the following options.

**URL to Consult Option:** Sets the list and order of the OCSP servers used to perform the revocation check. Select one of these values:

**UseAIAInCert:**

(Default) Use the URL of an online certificate status protocol server specified in the Authority Information Access (AIA) extension in the certificate. The AIA extension is used to identify how to access certificate authority (CA) information and services for the issuer of the certificate.

**LocalURL:**

Use the specified URL for the OCSP server specified in the OCSP Server URL option.

**UseAIAIfPresentElseLocal:**

Use the URL of the OCSP server specified in the AIA extension in the certificate if present. If the AIA extension is not present in the certificate, use the URL that is configured in the OCSP Server URL.

**UseAIAInSignerCert:**

Use the URL of the OCSP server specified in the AIA extension in the OCSP request of the signer certificate.

**OCSP Server URL:**

Sets the URL of the configured OCSP server. The value is used only when the LocalURL or UseAIAIfPresentElseLocal values are in URL To Consult Option.

**Revocation Check Style:**

Sets the revocation-checking style that is used for verifying the trust status of the CRL provider's certificate from its observed revocation status. Select one of these values:

Sets the URL of the configured OCSP server. The value is used only when the LocalURL or UseAIAIfPresentElseLocal values are in URL To Consult Option.

Revocation Check Style:

Sets the revocation-checking style that is used for verifying the trust status of the CRL provider's certificate from its observed revocation status. Select one of these values:

**NoCheck:**

Does not check for revocation.

**BestEffort:**

Checks for revocation of all certificates when possible.

**CheckIfAvailable:**

(Default) Checks for revocation of all certificates only when revocation information is available.

**AlwaysCheck:**

Checks for revocation of all certificates.

**Max Clock Skew Time (Minutes):**

Sets the maximum allowed skew, in minutes, between response time and local time. Valid skew times are `0` - `2147483647` min. The default value is `5` min.

**Response Freshness Time (Minutes):**

Sets the maximum time, in minutes, for which a preconstructed OCSP response is considered valid. Valid response freshness times are `1`- `2147483647` min. The default value is `525600` min. (one year).

**Send Nonce:**

Select this option to send a nonce with the OCSP request. A *nonce* is a parameter that varies with time. These parameters can be a timestamp, a visit counter on a web page, or a special marker. The parameter is intended to limit or prevent the unauthorized replay or reproduction of a file. When the option deselected, a nonce is not sent with the request. By default, the option is selected.

**Sign OCSP Request:**

Select this option to specify that the OCSP request must be signed. When the option is deselected, the OCSP request does not need be signed. By default, the option deselected.

**Request Signer Credential Alias:**

Sets the credential alias used for signing the OCSP request when signing is enabled.

**Go Online for OCSP:**

Select this option to access the network for OCSP information. The network can be accessed to retrieve OCSP information for OCSP checking. The AEM Forms Server uses embedded and cached OCSP information when possible to reduce the amount of network traffic generated due to OCSP checking. When the option is deselected, OCSP checking is not retrieved from the network, and only embedded and cached OCSP information is used. By default, the option is selected.

**Ignore Validity Dates:**

Select this option to use the OCSP response thisUpdate and nextUpdate times. Ignoring these response times prevents any negative effect on response validity. The thisUpdate and nextUpdate times are retrieved from external sources by using HTTP or LDAP, and can be different for each revocation information. When the option is deselected, the thisUpdate and nextUpdate times are ignored. By default, the option is deselected.

**Allow OCSP NoCheck Extension:**

Select this option to allow an OCSPNoCheck extension in the response signing certificate. An OCSP-NoCheck extension can be present in the OCSP Responder's certificate to prevent infinite loops from occurring during the validation process. When the option is deselected, the OCSPNoCheck extension is not used. By default, the option is selected.

**Require OCSP ISIS-MTT CertHash Extension:**

Select this option to specify that certificate public key hash (CertHash) extensions must be present in OCSP responses. This extension is required for SigQ validation. SigQ compliance requires the CertHash extension to be in the OCSP responder certificate. Select this option when processing for SigQ compliance and supported OCSP responders. When the option is deselected, the CertHash extension presence in the OCSP response is not required. By default, the option is deselected.

Sets the maximum allowed skew, in minutes, between response time and local time. Valid skew times are `0` - `2147483647` min. The default value is `5` min.

Response Freshness Time (Minutes):

Sets the maximum time, in minutes, for which a preconstructed OCSP response is considered valid. Valid response freshness times are `1`- `2147483647` min. The default value is `525600` min. (one year).

Send Nonce:

Select this option to send a nonce with the OCSP request. A *nonce* is a parameter that varies with time. These parameters can be a timestamp, a visit counter on a web page, or a special marker. The parameter is intended to limit or prevent the unauthorized replay or reproduction of a file. When the option deselected, a nonce is not sent with the request. By default, the option is selected.

Sign OCSP Request:

Select this option to specify that the OCSP request must be signed. When the option is deselected, the OCSP request does not need be signed. By default, the option deselected.

Request Signer Credential Alias:

Sets the credential alias used for signing the OCSP request when signing is enabled.

Go Online for OCSP:

Select this option to access the network for OCSP information. The network can be accessed to retrieve OCSP information for OCSP checking. The AEM Forms Server uses embedded and cached OCSP information when possible to reduce the amount of network traffic generated due to OCSP

checking. When the option is deselected, OCSP checking is not retrieved from the network, and only embedded and cached OCSP information is used. By default, the option is selected.

Ignore Validity Dates:

Select this option to use the OCSP response thisUpdate and nextUpdate times. Ignoring these response times prevents any negative effect on response validity. The thisUpdate and nextUpdate times are retrieved from external sources by using HTTP or LDAP, and can be different for each revocation information. When the option is deselected, the thisUpdate and nextUpdate times are ignored. By default, the option is deselected.

Allow OCSP NoCheck Extension:

Select this option to allow an OCSPNoCheck extension in the response signing certificate. An OCSP-NoCheck extension can be present in the OCSP Responder's certificate to prevent infinite loops from occurring during the validation process. When the option is deselected, the OCSPNoCheck extension is not used. By default, the option is selected.

Require OCSP ISIS-MTT CertHash Extension:

Select this option to specify that certificate public key hash (CertHash) extensions must be present in OCSP responses. This extension is required for SigQ validation. SigQ compliance requires the CertHash extension to be in the OCSP responder certificate. Select this option when processing for SigQ compliance and supported OCSP responders. When the option is deselected, the CertHash extension presence in the OCSP response is not required. By default, the option is deselected.

*CRL Options Spec*

(Optional) A `CRLOptionSpec` value that represents the certificate revocation list (CRL) preferences when CRL is used to perform revocation checking. If you provide a literal value, specify the following options.

**Consult Local URI First:**

Select this option to use the CRL location provided as a local URI before any specified locations within a certificate. The CRL location provided is used for revocation checking. When this option is selected, it means the local URI is used first. When this option is deselected, the locations specified in the certificate before using the local URI are used. By default, the option is deselected.

**Local URI for CRL Lookup:**

Sets the URL for the local CRL store. This value is used only if the Consult Local URI First option is selected. No default value is provided.

**Revocation Check Style:**

Sets the revocation-checking style used for verifying the trust status of the CRL provider's certificate from its observed revocation status. Select one of these values:

- **NoCheck:** Does not check for revocation.
- **BestEffort:** (Default) Checks for revocation of all certificates when possible.

- **CheckIfAvailable:** Checks for revocation of all certificates only when revocation information is available.
- **AlwaysCheck:** Checks for revocation of all certificates.

**LDAP Server:**

Sets the URL or path of the Lightweight Directory Access Protocol (LDAP) server used to retrieve information about the certificate revocation list (CRL). The LDAP server searches for CRL information using the distinguished name (DN) according to the rules specified in RFC 3280

, section 4.2.1.14. For example, you can type `www.ldap.com` for the URL or `ldap://ssl.ldap.com:200` for the path and port. No default value is provided.

**Go Online for CRL Retrieval:**

Select this option to access the network to retrieve CRL information. CRL information is cached on the server to improve network performance. CRL information is retrieved online only when necessary. When this option is deselected, CRL information is not retrieved online. By default, the option is selected.

**Ignore Validity Dates:**

Select this option to use thisUpdate and nextUpdate times. Ignoring the response's thisUpdate and nextUpdate times prevents any negative effect on response validity. The thisUpdate and nextUpdate times are retrieved from external sources by using HTTP or LDAP and can be different for each revocation information. When the option is deselected, the thisUpdate and nextUpdate time are ignored. By default, the option deselected.

**Require AKI Extension in CRL:**

Select this option to specify that the Authority Key Identifier (AKI) extension must be present in the CRL. The AKI extension can be used for CRL validation. When this option is deselected, the presence of the AKI extension the CRL is not required. By default, the option is deselected.

*TSP Options Spec*

(Optional) A `TSPOptionSpec` value that represents the settings that define timestamp information applied to the certified signature.

If you provide a literal value, specify the following options.

**Time Stamp Server URL:**

Sets the URL for a TSP server. If no value is provided, the timestamp from the local system is applied. No default value is provided.

**Time Stamp Server Username:**

Sets the user name, if necessary, for accessing the TSP server. No default value is provided.

**Time Stamp Server Password:**

Sets the password for the user name, if necessary, for accessing the TSP server. No default value is provided.

**Time Stamp Server Hash Algorithm:**

Sets the hash algorithm used to digest the request sent to the timestamp provider. Select one of these values:

**SHA1: (Default)**

The Secure Hash Algorithm that has a 160-bit hash value.

**SHA256:**

The Secure Hash Algorithm that has a 256-bit hash value.

**SHA384:**

The Secure Hash Algorithm that has a 384-bit hash value.

**SHA512:**

The Secure Hash Algorithm that has a 512-bit hash value.

**RIPEMD160:**

The RACE Integrity Primitives Evaluation Message Digest that has a 160-bit message digest algorithm and is not FIPS-compliant.

**Predicted Time Stamp Token Size (In Bytes):**

Sets the estimated size, in bytes, of the TSP response. The size is used to create a signature hole in the PDF document. This value represents the maximum size of the timestamp response that the configured TSP could return. Configuring an undersized value can cause the operation to fail; however, configuring an oversized value causes the size to be larger than necessary. It is recommended that this value is not modified unless the timestamp server requires a response size to be less than 4096 bytes. Valid values are from `60` to `10240`. The default value is `4096`.

**Send Nonce:**

Select this option to send a nonce with the request. A *nonce* is a parameter that varies with time. These parameters can be a timestamp, a visit counter on a web page, or a special marker. The parameter is intended to limit or prevent the unauthorized replay or reproduction of a file. When the option deselected, a nonce is not sent with the request. By default, the option is selected.

**Output properties**

Property to specify the output PDF document.

*Output PDF*

The location in the process data model to store the signed PDF document. This document is new, and the input PDF document is not modified. The data type is `document`.

### Exceptions

This operation can throw PDFOperationException, SigningException, PermissionsException, InvalidArgumentException, SeedValueValidationException, MissingSignatureFieldException, SignatureFieldSignedException, CredentialLoginException, and FIPSComplianceException exceptions.

## Verify PDF Document operation

Verifies the signatures and returns information regarding the overall validity of a PDF document. The validity of a PDF document includes the signed content and the identity and trust settings of the signer. The trust settings that are used are configured in Trust Store. To verify the identities of signers, add the signers to Trust Store. (See Trust Store ManagementHelp

.) Document validity determines whether MDP (Modification Detection and Prevention) and MDP+ (Modification Detection and Protection Plus) rules are adhered to.

For example, your application must verify the identity of people who signed the PDF document and validate that the PDF document has not been altered since it was signed. You use the PDF Document operation to validate that the signatures are trusted and verify that fields locked by MDP+ have not been modified.

For information about the General and Route Evaluation property groups, see Commonoperationproperties.

### Common properties

Properties to specify the input PDF document.

*Input PDF*

A `document` value that represents the PDF document to verify.

If you provide a literal value, clicking the ellipsis button opens the Select Asset dialog box. (See AboutSelectAsset.)

### PKI Options properties

Properties to specify revocation-checking style, verification time, path validation, time-stamping, certification revocation lists, and online certificate status protocol settings for verifying the PDF document.

*Revocation Check Style*

A `RevocationCheckStyle` value that specifies the revocation-checking style used for verifying the trust status of the CRL provider's certificate from its observed revocation status.

If you provide a literal value, select one of these values:

**NoCheck:**

Does not check for revocation.

**BestEffort:**

Checks for revocation of all certificates when possible.

**CheckIfAvailable:**

(Default) Checks for revocation of all certificates only when revocation information is available.

**AlwaysCheck:**

Checks for revocation of all certificates.

*Verification Time*

(Optional) A `VerificationTime` value that specifies the verification time to use. The default value is Secure Time Else Current Time.

If you provide a literal value, select one of these values:

**Signing Time:**

The time that the signature was applied as given by the signer's computer.

**Current Time:**

The time that the verification operation is being carried out.

**Secure Time Else Current Time:**

The time specified by a trusted time-stamping authority.

If you specify Secure Time Else Current Time and validation returns a status of unknown with a trusted timestamp, the validation is checked using Current Time.

*OCSP Options Spec*

(Optional) An `OCSPOptionSpec` value that represents settings for using Online Certificate Status Protocol (OCSP) revocation checking. To provide a literal value, specify the following options.

**URL to Consult Option:** Sets the list and order of the OCSP servers used to perform the revocation check. Select one of these values:

**UseAIAInCert:**

(Default) Use the URL of an online certificate status protocol server specified in the Authority Information Access (AIA) extension in the certificate. The AIA extension is used to identify how to access certificate authority (CA) information and services for the issuer of the certificate.

**LocalURL:**

Use the specified URL for the OCSP server specified in the OCSP Server URL option.

**UseAIAIfPresentElseLocal:**

Use the URL of the OCSP server specified in the AIA extension in the certificate if present. If the AIA extension is not present in the certificate, use the URL that is configured in the OCSP Server URL.

**UseAIAInSignerCert:**

Use the URL of the OCSP server specified in the AIA extension in the OCSP request of the signer certificate.

**OCSP Server URL:**

Sets the URL of the configured OCSP server. The value is used only when the LocalURL or UseAIAIf-PresentElseLocal values are in URL To Consult Option.

**Revocation Check Style:**

Sets the revocation-checking style that is used for verifying the trust status of the CRL provider's certificate from its observed revocation status. Select one of these values:

Sets the URL of the configured OCSP server. The value is used only when the LocalURL or UseAIAIf-PresentElseLocal values are in URL To Consult Option.

Revocation Check Style:

Sets the revocation-checking style that is used for verifying the trust status of the CRL provider's certificate from its observed revocation status. Select one of these values:

**NoCheck:**

Does not check for revocation.

**BestEffort:**

Checks for revocation of all certificates when possible.

**CheckIfAvailable:**

(Default) Checks for revocation of all certificates only when revocation information is available.

**AlwaysCheck:**

Checks for revocation of all certificates.

**Max Clock Skew Time (Minutes):**

Sets the maximum allowed skew, in minutes, between response time and local time. Valid skew times are `0` - `2147483647` min. The default value is `5` min.

**Response Freshness Time (Minutes):**

Sets the maximum time, in minutes, for which a preconstructed OCSP response is considered valid. Valid response freshness times are `1`- `2147483647` min. The default value is `525600` min. (one year).

**Send Nonce:**

Select this option to send a nonce with the OCSP request. A *nonce* is a parameter that varies with time. These parameters can be a timestamp, a visit counter on a web page, or a special marker. The

parameter is intended to limit or prevent the unauthorized replay or reproduction of a file. When the option deselected, a nonce is not sent with the request. By default, the option is selected.

**Sign OCSP Request:**

Select this option to specify that the OCSP request must be signed. When the option is deselected, the OCSP request does not need be signed. By default, the option deselected.

**Request Signer Credential Alias:**

Sets the credential alias used for signing the OCSP request when signing is enabled.

**Go Online for OCSP:**

Select this option to access the network for OCSP information. The network can be accessed to retrieve OCSP information for OCSP checking. The AEM Forms Server uses embedded and cached OCSP information when possible to reduce the amount of network traffic generated due to OCSP checking. When the option is deselected, OCSP checking is not retrieved from the network, and only embedded and cached OCSP information is used. By default, the option is selected.

**Ignore Validity Dates:**

Select this option to use the OCSP response thisUpdate and nextUpdate times. Ignoring these response times prevents any negative effect on response validity. The thisUpdate and nextUpdate times are retrieved from external sources by using HTTP or LDAP, and can be different for each revocation information. When the option is deselected, the thisUpdate and nextUpdate times are ignored. By default, the option is deselected.

**Allow OCSP NoCheck Extension:**

Select this option to allow an OCSPNoCheck extension in the response signing certificate. An OCSP-NoCheck extension can be present in the OCSP Responder's certificate to prevent infinite loops from occurring during the validation process. When the option is deselected, the OCSPNoCheck extension is not used. By default, the option is selected.

**Require OCSP ISIS-MTT CertHash Extension:**

Select this option to specify that certificate public key hash (CertHash) extensions must be present in OCSP responses. This extension is required for SigQ validation. SigQ compliance requires the CertHash extension to be in the OCSP responder certificate. Select this option when processing for SigQ compliance and supported OCSP responders. When the option is deselected, the CertHash extension presence in the OCSP response is not required. By default, the option is deselected.

Sets the maximum allowed skew, in minutes, between response time and local time. Valid skew times are `0` - `2147483647` min. The default value is `5` min.

Response Freshness Time (Minutes):

Sets the maximum time, in minutes, for which a preconstructed OCSP response is considered valid. Valid response freshness times are `1`- `2147483647` min. The default value is `525600` min. (one year).

Send Nonce:

Select this option to send a nonce with the OCSP request. A *nonce* is a parameter that varies with time. These parameters can be a timestamp, a visit counter on a web page, or a special marker. The parameter is intended to limit or prevent the unauthorized replay or reproduction of a file. When the option deselected, a nonce is not sent with the request. By default, the option is selected.

Sign OCSP Request:

Select this option to specify that the OCSP request must be signed. When the option is deselected, the OCSP request does not need be signed. By default, the option deselected.

Request Signer Credential Alias:

Sets the credential alias used for signing the OCSP request when signing is enabled.

Go Online for OCSP:

Select this option to access the network for OCSP information. The network can be accessed to retrieve OCSP information for OCSP checking. The AEM Forms Server uses embedded and cached OCSP information when possible to reduce the amount of network traffic generated due to OCSP checking. When the option is deselected, OCSP checking is not retrieved from the network, and only embedded and cached OCSP information is used. By default, the option is selected.

Ignore Validity Dates:

Select this option to use the OCSP response thisUpdate and nextUpdate times. Ignoring these response times prevents any negative effect on response validity. The thisUpdate and nextUpdate times are retrieved from external sources by using HTTP or LDAP, and can be different for each revocation information. When the option is deselected, the thisUpdate and nextUpdate times are ignored. By default, the option is deselected.

Allow OCSP NoCheck Extension:

Select this option to allow an OCSPNoCheck extension in the response signing certificate. An OCSP-NoCheck extension can be present in the OCSP Responder's certificate to prevent infinite loops from occurring during the validation process. When the option is deselected, the OCSPNoCheck extension is not used. By default, the option is selected.

Require OCSP ISIS-MTT CertHash Extension:

Select this option to specify that certificate public key hash (CertHash) extensions must be present in OCSP responses. This extension is required for SigQ validation. SigQ compliance requires the CertHash extension to be in the OCSP responder certificate. Select this option when processing for SigQ compliance and supported OCSP responders. When the option is deselected, the CertHash extension presence in the OCSP response is not required. By default, the option is deselected.

*CRL Options Spec*

(Optional) A `CRLOptionSpec` value that represents the certificate revocation list (CRL) preferences when CRL is used to perform revocation checking. If you provide a literal value, specify the following options.

**Consult Local URI First:**

Select this option to use the CRL location provided as a local URI before any specified locations within a certificate. The CRL location provided is used for revocation checking. When this option is selected, it means the local URI is used first. When this option is deselected, the locations specified in the certificate before using the local URI are used. By default, the option is deselected.

**Local URI for CRL Lookup:**

Sets the URL for the local CRL store. This value is used only if the Consult Local URI First option is selected. No default value is provided.

**Revocation Check Style:**

Sets the revocation-checking style used for verifying the trust status of the CRL provider's certificate from its observed revocation status. Select one of these values:

- **NoCheck:** Does not check for revocation.
- **BestEffort:** (Default) Checks for revocation of all certificates when possible.
- **CheckIfAvailable:** Checks for revocation of all certificates only when revocation information is available.
- **AlwaysCheck:** Checks for revocation of all certificates.

**LDAP Server:**

Sets the URL or path of the Lightweight Directory Access Protocol (LDAP) server used to retrieve information about the certificate revocation list (CRL). The LDAP server searches for CRL information using the distinguished name (DN) according to the rules specified in RFC 3280

, section 4.2.1.14. For example, you can type `www.ldap.com` for the URL or `ldap://ssl.ldap.com:200` for the path and port. No default value is provided.

**Go Online for CRL Retrieval:**

Select this option to access the network to retrieve CRL information. CRL information is cached on the server to improve network performance. CRL information is retrieved online only when necessary. When this option is deselected, CRL information is not retrieved online. By default, the option is selected.

**Ignore Validity Dates:**

Select this option to use thisUpdate and nextUpdate times. Ignoring the response's thisUpdate and nextUpdate times prevents any negative effect on response validity. The thisUpdate and nextUpdate times are retrieved from external sources by using HTTP or LDAP and can be different for each revocation information. When the option is deselected, the thisUpdate and nextUpdate time are ignored. By default, the option deselected.

**Require AKI Extension in CRL:**

Select this option to specify that the Authority Key Identifier (AKI) extension must be present in the CRL. The AKI extension can be used for CRL validation. When this option is deselected, the presence of the AKI extension the CRL is not required. By default, the option is deselected.

*Path Validation Options Spec*

(Optional) A `PathValidationOptionSpec` that represents the settings that control RFC3280-related path validation options. For example, you can indicate whether policy mapping is allowed in the certification path. (See RFC3280

). If you provide a literal value, you can set the following options.

**Require Explicit Policy:**

Select this option to specify that the path must be valid for at least one of the certificate policies in the user initial policy set. When this option is deselected, the path validity is not required. By default, the option is deselected.

**Inhibit Any Policy:**

Select this option to specify that a policy object identifier (OID) must be processed if it is included in a certificate. When deselected, any policy can be selected. By default, the option is deselected.

**Check All Paths:**

Select this option to require that all paths to a trust anchor must be validated. When this option is deselected, all paths to a trust anchor are not validated. By default, the option is deselected.

**Inhibit Policy Mapping:**

Select this option to allow policy mapping in the certification path. When this option is deselect, policy mapping is not allowed in the certification path. By default, the option is deselected.

**LDAP Server:**

Sets the URL or path of the Lightweight Directory Access Protocol (LDAP) server used to retrieve information about the certificate revocation list (CRL). The LDAP server searches for CRL information using the distinguished name (DN) according to the rules specified in RFC 3280

, section 4.2.1.14. For example, you can type `www.ldap.com` for the URL or `ldap://ssl.ldap.com:200` for the path and port. No default value is provided.

**Follow URIs in Certificate AIA:**

Select this option to specify to follow any URIs specified in the certificate’s Authority Information Access (AIA) extension for path discovery. The AIA extension specifies where to find up-to-date certificates. When this option is deselected, no URIs are processed in the AIA extension from the certificate. By default, the option is deselected.

**Basic Constraints Extension Required in CA Certificates:**

Select this option to specify that the certificate authority (CA) Basic Constraints certificate extension must be present for CA certificates. Some early German certified root certificates (7 and earlier) are not compliant to RFC3280

and do not contain the basic constraint extension. If it is known that a user's EE certificate chains up to such a German root, deselect this option. When this option is deselected, the presence of the CA Basic Constraints certificate in CA certificates is not required. By default, the value is selected.

**Require Valid Certificate Signature During Chain Building:**

Select this option to require that all Digital Signature Algorithm (DSA) signatures on certificates be valid before a chain is built. For example, in a chain CA > ICA > EE where the signature for EE is not valid, the chain building stops at ICA. EEs are not included in the chain. When this option is deselected, the entire chain is built regardless of whether an invalid DSA signature is encountered. By default, the option is deselected.

*TSP Options Spec*

(Optional) A `TSPOptionSpec` value that represents the settings that define time-stamping information applied to the certified signature. Only the Revocation Check Style (`tspRevocationCheckStyle` data item) option is used from the `TSPOptionSpec` value.

If you provide a literal value, specify the following option.

**Revocation Check Style:**

A list that specifies the revocation-checking style used for verifying the trust status of the CRL provider's certificate from its observed revocation status. Select one of these values:

**NoCheck:**

Does not check for revocation.

**BestEffort:**

(Default) Checks for revocation of all certificates when possible.

**CheckIfAvailable:**

Checks for revocation of all certificates only when revocation information is available.

**AlwaysCheck:**

Checks for revocation of all certificates.

**Use Expired Timestamps:**

Select this option to use timestamps that have expired during the validation of the certificate. When this option is deselected, expired timestamps are not used. By default, this option is selected.

**SPI Options properties**

Properties to specify the name and properties passed to a custom SPI for verifying the PDF signature. For example, you use a custom signature handler. The custom signature handler can reference a security credential stored in locations that are accessible over a network instead of the AEM Forms trust store. For information about creating custom service providers, see Programming withAEM formsES2.5

.

*Name of SPI Service*

(Optional) A `string` value that specifies the name of the SPI.

*Properties Map To Be Passed To SPI*

(Optional) A `map` of `string` values that specifies the properties passed to the SPI to verify the certificate.

If you provide a literal value, clicking the ellipsis button opens the SPI Properties dialog box. (See AboutSPIProperties.)

The file you choose must contain a property-value pair. Each property-value pair must be formatted as *[property name]=[value]*, where *[property name]* is the name of the property and *[value]* is the value assigned to the property. The design of the SPI determines the property- value pairs that is used.

**Output properties**

*PDF Document Verification Info*

The location in the process data model to store the verification information. The verification information contains information about the signature and its validity status. The data type is `PDFDocumentVerificationInfo`.

**Exceptions**

This operation can throw PDFOperationException, InvalidArgumentException, and PermissionsException exceptions.

## Verify PDF Signature operation

Verifies the signature in a signature field and returns information about the signature. Verification includes the signed content and the identity of the signer. After the verification occurs, an error occurs to indicate that the document was signed in the future. The error occurs because the signature, which is being verified, was applied at a time that is more than 65 min. You can change this setting in administration console. (See Applications andServices administrationhelp

) The trust settings that are used are configured in Trust Store. Before identities of signers can be verified, they must be added in Trust Store. (See Trust Store ManagementHelp

.)

For example, your application must verify the identity and trust chain of people who signed the PDF document. You use the Verify PDF Signature operation to verify that the identity and trust chain of the signer.

For information about the General and Route Evaluation property groups, see Commonoperationproperties.

**Common properties**

Properties that specify the input PDF document and verification information.

*Input PDF*

A `document` value that represents a PDF document that contains a signature to verify.

If you provide a literal value, clicking the ellipsis button opens the Select Asset dialog box. (See AboutSelectAsset.)

When you provide a PDF document that has signed signature fields, it populates the Signature Field Name property as a list. The list contains fully qualified names of the signed signature fields in the PDF document.

*Signature Field Name*

A `string` value that represents the name of the signature field that contains a signature to verify. Use a fully qualified name for the signature field. When using a PDF document based on a form created in Designer, the partial name of the signature field can be used. For example, `form1[0].#subform[1].SignatureField3[3]` can be specified as `SignatureField3[3]`.

If you provide a literal value for the Signature Field Name property and a literal value is provided in the Input PDF property, a list appears. Select one of the values from the list of fully qualified names. Each fully qualified name represents a signed signature field in the provided PDF document.

**PKI Options properties**

Properties to specify revocation-checking style, verification time, path validation, time-stamping, certification revocation lists, and online certificate status protocol settings for verifying the PDF document.

*Revocation Check Style*

A `RevocationCheckStyle` value that specifies the revocation-checking style used for verifying the trust status of the CRL provider's certificate from its observed revocation status.

If you provide a literal value, select one of these values:

**NoCheck:**

Does not check for revocation.

**BestEffort:**

Checks for revocation of all certificates when possible.

**CheckIfAvailable:**

(Default) Checks for revocation of all certificates only when revocation information is available.

**AlwaysCheck:**

Checks for revocation of all certificates.

*Verification Time*

(Optional) A `VerificationTime` value that specifies the verification time to use. The default value is Secure Time Else Current Time.

If you provide a literal value, select one of these values:

**Signing Time:**

The time that the signature was applied as given by the signer's computer.

**Current Time:**

The time that the verification operation is being carried out.

**Secure Time Else Current Time:**

The time specified by a trusted time-stamping authority.

If you specify Secure Time Else Current Time and validation returns a status of unknown with a trusted timestamp, the validation is checked using Current Time.

*OCSP Options Spec*

(Optional) An `OCSPOptionSpec` value that represents settings for using Online Certificate Status Protocol (OCSP) revocation checking. To provide a literal value, specify the following options.

**URL to Consult Option:** Sets the list and order of the OCSP servers used to perform the revocation check. Select one of these values:

**UseAIAInCert:**

(Default) Use the URL of an online certificate status protocol server specified in the Authority Information Access (AIA) extension in the certificate. The AIA extension is used to identify how to access certificate authority (CA) information and services for the issuer of the certificate.

**LocalURL:**

Use the specified URL for the OCSP server specified in the OCSP Server URL option.

**UseAIAIfPresentElseLocal:**

Use the URL of the OCSP server specified in the AIA extension in the certificate if present. If the AIA extension is not present in the certificate, use the URL that is configured in the OCSP Server URL.

**UseAIAInSignerCert:**

Use the URL of the OCSP server specified in the AIA extension in the OCSP request of the signer certificate.

**OCSP Server URL:**

Sets the URL of the configured OCSP server. The value is used only when the LocalURL or UseAIAIfPresentElseLocal values are in URL To Consult Option.

**Revocation Check Style:**

Sets the revocation-checking style that is used for verifying the trust status of the CRL provider’s certificate from its observed revocation status. Select one of these values:

Sets the URL of the configured OCSP server. The value is used only when the LocalURL or UseAIAIfPresentElseLocal values are in URL To Consult Option.

Revocation Check Style:

Sets the revocation-checking style that is used for verifying the trust status of the CRL provider's certificate from its observed revocation status. Select one of these values:

**NoCheck:**

Does not check for revocation.

**BestEffort:**

Checks for revocation of all certificates when possible.

**CheckIfAvailable:**

(Default) Checks for revocation of all certificates only when revocation information is available.

**AlwaysCheck:**

Checks for revocation of all certificates.

**Max Clock Skew Time (Minutes):**

Sets the maximum allowed skew, in minutes, between response time and local time. Valid skew times are `0` - `2147483647` min. The default value is `5` min.

**Response Freshness Time (Minutes):**

Sets the maximum time, in minutes, for which a preconstructed OCSP response is considered valid. Valid response freshness times are `1`- `2147483647` min. The default value is `525600` min. (one year).

**Send Nonce:**

Select this option to send a nonce with the OCSP request. A *nonce* is a parameter that varies with time. These parameters can be a timestamp, a visit counter on a web page, or a special marker. The parameter is intended to limit or prevent the unauthorized replay or reproduction of a file. When the option deselected, a nonce is not sent with the request. By default, the option is selected.

**Sign OCSP Request:**

Select this option to specify that the OCSP request must be signed. When the option is deselected, the OCSP request does not need be signed. By default, the option deselected.

**Request Signer Credential Alias:**

Sets the credential alias used for signing the OCSP request when signing is enabled.

**Go Online for OCSP:**

Select this option to access the network for OCSP information. The network can be accessed to retrieve OCSP information for OCSP checking. The AEM Forms Server uses embedded and cached OCSP information when possible to reduce the amount of network traffic generated due to OCSP checking. When the option is deselected, OCSP checking is not retrieved from the network, and only embedded and cached OCSP information is used. By default, the option is selected.

**Ignore Validity Dates:**

Select this option to use the OCSP response thisUpdate and nextUpdate times. Ignoring these response times prevents any negative effect on response validity. The thisUpdate and nextUpdate times are retrieved from external sources by using HTTP or LDAP, and can be different for each revocation information. When the option is deselected, the thisUpdate and nextUpdate times are ignored. By default, the option is deselected.

**Allow OCSP NoCheck Extension:**

Select this option to allow an OCSPNoCheck extension in the response signing certificate. An OCSP-NoCheck extension can be present in the OCSP Responder's certificate to prevent infinite loops from occurring during the validation process. When the option is deselected, the OCSPNoCheck extension is not used. By default, the option is selected.

**Require OCSP ISIS-MTT CertHash Extension:**

Select this option to specify that certificate public key hash (CertHash) extensions must be present in OCSP responses. This extension is required for SigQ validation. SigQ compliance requires the CertHash extension to be in the OCSP responder certificate. Select this option when processing for SigQ compliance and supported OCSP responders. When the option is deselected, the CertHash extension presence in the OCSP response is not required. By default, the option is deselected.

Sets the maximum allowed skew, in minutes, between response time and local time. Valid skew times are `0` - `2147483647` min. The default value is `5` min.

Response Freshness Time (Minutes):

Sets the maximum time, in minutes, for which a preconstructed OCSP response is considered valid. Valid response freshness times are `1`- `2147483647` min. The default value is `525600` min. (one year).

Send Nonce:

Select this option to send a nonce with the OCSP request. A *nonce* is a parameter that varies with time. These parameters can be a timestamp, a visit counter on a web page, or a special marker. The parameter is intended to limit or prevent the unauthorized replay or reproduction of a file. When the option deselected, a nonce is not sent with the request. By default, the option is selected.

Sign OCSP Request:

Select this option to specify that the OCSP request must be signed. When the option is deselected, the OCSP request does not need be signed. By default, the option deselected.

Request Signer Credential Alias:

Sets the credential alias used for signing the OCSP request when signing is enabled.

Go Online for OCSP:

Select this option to access the network for OCSP information. The network can be accessed to retrieve OCSP information for OCSP checking. The AEM Forms Server uses embedded and cached OCSP information when possible to reduce the amount of network traffic generated due to OCSP

checking. When the option is deselected, OCSP checking is not retrieved from the network, and only embedded and cached OCSP information is used. By default, the option is selected.

Ignore Validity Dates:

Select this option to use the OCSP response thisUpdate and nextUpdate times. Ignoring these response times prevents any negative effect on response validity. The thisUpdate and nextUpdate times are retrieved from external sources by using HTTP or LDAP, and can be different for each revocation information. When the option is deselected, the thisUpdate and nextUpdate times are ignored. By default, the option is deselected.

Allow OCSP NoCheck Extension:

Select this option to allow an OCSPNoCheck extension in the response signing certificate. An OCSP-NoCheck extension can be present in the OCSP Responder's certificate to prevent infinite loops from occurring during the validation process. When the option is deselected, the OCSPNoCheck extension is not used. By default, the option is selected.

Require OCSP ISIS-MTT CertHash Extension:

Select this option to specify that certificate public key hash (CertHash) extensions must be present in OCSP responses. This extension is required for SigQ validation. SigQ compliance requires the CertHash extension to be in the OCSP responder certificate. Select this option when processing for SigQ compliance and supported OCSP responders. When the option is deselected, the CertHash extension presence in the OCSP response is not required. By default, the option is deselected.

*CRL Options Spec*

(Optional) A `CRLOptionSpec` value that represents the certificate revocation list (CRL) preferences when CRL is used to perform revocation checking. If you provide a literal value, specify the following options.

**Consult Local URI First:**

Select this option to use the CRL location provided as a local URI before any specified locations within a certificate. The CRL location provided is used for revocation checking. When this option is selected, it means the local URI is used first. When this option is deselected, the locations specified in the certificate before using the local URI are used. By default, the option is deselected.

**Local URI for CRL Lookup:**

Sets the URL for the local CRL store. This value is used only if the Consult Local URI First option is selected. No default value is provided.

**Revocation Check Style:**

Sets the revocation-checking style used for verifying the trust status of the CRL provider's certificate from its observed revocation status. Select one of these values:

- **NoCheck:** Does not check for revocation.
- **BestEffort:** (Default) Checks for revocation of all certificates when possible.

- **CheckIfAvailable:** Checks for revocation of all certificates only when revocation information is available.
- **AlwaysCheck:** Checks for revocation of all certificates.

**LDAP Server:**

Sets the URL or path of the Lightweight Directory Access Protocol (LDAP) server used to retrieve information about the certificate revocation list (CRL). The LDAP server searches for CRL information using the distinguished name (DN) according to the rules specified in RFC 3280

, section 4.2.1.14. For example, you can type `www.ldap.com` for the URL or `ldap://ssl.ldap.com:200` for the path and port. No default value is provided.

**Go Online for CRL Retrieval:**

Select this option to access the network to retrieve CRL information. CRL information is cached on the server to improve network performance. CRL information is retrieved online only when necessary. When this option is deselected, CRL information is not retrieved online. By default, the option is selected.

**Ignore Validity Dates:**

Select this option to use thisUpdate and nextUpdate times. Ignoring the response's thisUpdate and nextUpdate times prevents any negative effect on response validity. The thisUpdate and nextUpdate times are retrieved from external sources by using HTTP or LDAP and can be different for each revocation information. When the option is deselected, the thisUpdate and nextUpdate time are ignored. By default, the option deselected.

**Require AKI Extension in CRL:**

Select this option to specify that the Authority Key Identifier (AKI) extension must be present in the CRL. The AKI extension can be used for CRL validation. When this option is deselected, the presence of the AKI extension the CRL is not required. By default, the option is deselected.

*Path Validation Options Spec*

(Optional) A `PathValidationOptionSpec` that represents the settings that control RFC3280-related path validation options. For example, you can indicate whether policy mapping is allowed in the certification path. (See RFC3280

). If you provide a literal value, you can set the following options.

**Require Explicit Policy:**

Select this option to specify that the path must be valid for at least one of the certificate policies in the user initial policy set. When this option is deselected, the path validity is not required. By default, the option is deselected.

**Inhibit Any Policy:**

Select this option to specify that a policy object identifier (OID) must be processed if it is included in a certificate. When deselected, any policy can be selected. By default, the option is deselected.

**Check All Paths:**

Select this option to require that all paths to a trust anchor must be validated. When this option is deselected, all paths to a trust anchor are not validated. By default, the option is deselected.

**Inhibit Policy Mapping:**

Select this option to allow policy mapping in the certification path. When this option is deselect, policy mapping is not allowed in the certification path. By default, the option is deselected.

**LDAP Server:**

Sets the URL or path of the Lightweight Directory Access Protocol (LDAP) server used to retrieve information about the certificate revocation list (CRL). The LDAP server searches for CRL information using the distinguished name (DN) according to the rules specified in RFC 3280

, section 4.2.1.14. For example, you can type `www.ldap.com` for the URL or `ldap://ssl.ldap.com:200` for the path and port. No default value is provided.

**Follow URIs in Certificate AIA:**

Select this option to specify to follow any URIs specified in the certificate's Authority Information Access (AIA) extension for path discovery. The AIA extension specifies where to find up-to-date certificates. When this option is deselected, no URIs are processed in the AIA extension from the certificate. By default, the option is deselected.

**Basic Constraints Extension Required in CA Certificates:**

Select this option to specify that the certificate authority (CA) Basic Constraints certificate extension must be present for CA certificates. Some early German certified root certificates (7 and earlier) are not compliant to RFC3280

and do not contain the basic constraint extension. If it is known that a user's EE certificate chains up to such a German root, deselect this option. When this option is deselected, the presence of the CA Basic Constraints certificate in CA certificates is not required. By default, the value is selected.

**Require Valid Certificate Signature During Chain Building:**

Select this option to require that all Digital Signature Algorithm (DSA) signatures on certificates be valid before a chain is built. For example, in a chain CA > ICA > EE where the signature for EE is not valid, the chain building stops at ICA. EEs are not included in the chain. When this option is deselected, the entire chain is built regardless of whether an invalid DSA signature is encountered. By default, the option is deselected.

*TSP Options Spec*

(Optional) A `TSPOptionSpec` value that represents the settings that define time-stamping information applied to the certified signature. Only the Revocation Check Style (`tspRevocationCheckStyle` data item) option is used from the `TSPOptionSpec` value.

If you provide a literal value, specify the following option.

**Revocation Check Style:**

A list that specifies the revocation-checking style used for verifying the trust status of the CRL provider's certificate from its observed revocation status. Select one of these values:

**NoCheck:**

Does not check for revocation.

**BestEffort:**

(Default) Checks for revocation of all certificates when possible.

**CheckIfAvailable:**

Checks for revocation of all certificates only when revocation information is available.

**AlwaysCheck:**

Checks for revocation of all certificates.

**Use Expired Timestamps:**

Select this option to use timestamps that have expired during the validation of the certificate. When this option is deselected, expired timestamps are not used. By default, this option is selected.

**SPI Options properties**

Properties to specify the name and properties passed to a custom SPI for verifying the PDF signature. For example, you use a custom signature handler. The custom signature handler can reference a security credential stored in locations that are accessible over a network instead of the AEM Forms trust store. For information about creating custom service providers, see Programming withAEM formsES2.5

.

*Name of SPI Service*

(Optional) A `string` value that specifies the name of the SPI.

*Properties Map To Be Passed To SPI*

(Optional) A `map` of `string` values that specifies the properties passed to the SPI to verify the certificate.

If you provide a literal value, clicking the ellipsis button opens the SPI Properties dialog box. (See AboutSPIProperties.)

The file you choose must contain a property-value pair. Each property-value pair must be formatted as *[property name]=[value]*, where *[property name]* is the name of the property and *[value]* is the value assigned to the property. The design of the SPI determines the property- value pairs that is used.

**Output properties**

Property to specify the verification result.

*PDF Signature Verification Result*

The location in the process data model to store information about the signature and its validity status. The data type is `PDFSignatureVerificationInfo`.

**Exceptions**

This operation can throw PDFOperationException, MissingSignatureFieldException, InvalidArgumentException, SignatureFieldNotSignedException, SignatureVerifyException, and PermissionsException exceptions.

## Verify PDF Signature operation (deprecated)

**NOTE:** This operation is deprecated in AEM Forms version 8.2 and later. Use the VerifyPDFSignature operation instead. It is recommended that when you upgrade a process, you change it to use the Verify PDF Signature operation. (See Aboutdeprecatedoperations.)

Verifies the signature in a signature field and returns information about the signature. After the verification occurs, an error occurs to indicate that the document was signed in the future. The error occurs because the signature, which is being verified, was applied at a time that is more than 65 min. You can change this setting in administration console. (See Applications andServices AdministrationHelp

).

For information about the General and Route Evaluation property groups, see Commonoperationproperties.

**Common properties**

Properties to specify the input PDF document and various verification information.

*Input PDF*

A `document` value that represents a PDF document that contains a signature to verify.

If you provide a literal value, clicking the ellipsis button opens the Select Asset dialog box. (See AboutSelectAsset.)

When you provide a PDF document that has signed signature fields, it populates the Signature Field Name property as a list. The list contains fully qualified names of the signed signature fields in the PDF document.

*Signature Field Name*

A `string` value that represents the name of the signature field that contains a signature to verify. The fully qualified name of the signature field can be specified. When using a PDF document based on a form created in Designer, the partial name of the signature field can be used. For example, `form1[0].#subform[1].SignatureField3[3]` can be specified as `SignatureField3[3]`.

If you are verifying a PDF document, the partial name of the signature field can also be used. For example, `form1[0].#subform[1].SignatureField3[3]` can be specified as `SignatureField3[3]`. If

multiple signature fields exist with a similar partial name, the first signature field enumerated with the same partial name is signed. It is recommended that a fully qualified name is used to avoid these situations.

If you provide a literal value for the Signature Field Name property and a literal value is provided in the Input PDF property, a list appears. Select one of the values from the list of fully qualified names. Each fully qualified name represents a signed signature field in the provided PDF document.

*Revocation Check Style*

A `RevocationCheckStyle` value that specifies the revocation-checking style used for verifying the trust status of the CRL provider's certificate from its observed revocation status.

If you provide a literal value, select one of these values:

**NoCheck:**

Does not check for revocation.

**BestEffort:**

Checks for revocation of all certificates when possible.

**CheckIfAvailable:**

(Default) Checks for revocation of all certificates only when revocation information is available.

**AlwaysCheck:**

Checks for revocation of all certificates.

*Verification Time*

(Optional) A `VerificationTime` value that specifies the verification time to use. The default value is Secure Time Else Current Time.

If you provide a literal value, select one of these values:

**Signing Time:**

The time that the signature was applied as given by the signer's computer.

**Current Time:**

The time that the verification operation is being carried out.

**Secure Time Else Current Time:**

The time specified by a trusted time-stamping authority.

If you specify Secure Time Else Current Time and validation returns a status of unknown with a trusted timestamp, the validation is checked using Current Time.

**Advanced properties**

Properties to specify path validation and other advanced settings.

*Path Validation Options Spec*

(Optional) A `PathValidationOptionSpec` that represents the settings that control RFC3280-related path validation options. For example, you can indicate whether policy mapping is allowed in the certification path. (See RFC3280

). If you provide a literal value, you can set the following options.

**Require Explicit Policy:**

Select this option to specify that the path must be valid for at least one of the certificate policies in the user initial policy set. When this option is deselected, the path validity is not required. By default, the option is deselected.

**Inhibit Any Policy:**

Select this option to specify that a policy object identifier (OID) must be processed if it is included in a certificate. When deselected, any policy can be selected. By default, the option is deselected.

**Check All Paths:**

Select this option to require that all paths to a trust anchor must be validated. When this option is deselected, all paths to a trust anchor are not validated. By default, the option is deselected.

**Inhibit Policy Mapping:**

Select this option to allow policy mapping in the certification path. When this option is deselect, policy mapping is not allowed in the certification path. By default, the option is deselected.

**LDAP Server:**

Sets the URL or path of the Lightweight Directory Access Protocol (LDAP) server used to retrieve information about the certificate revocation list (CRL). The LDAP server searches for CRL information using the distinguished name (DN) according to the rules specified in RFC 3280

, section 4.2.1.14. For example, you can type `www.ldap.com` for the URL or `ldap://ssl.ldap.com:200` for the path and port. No default value is provided.

**Follow URIs in Certificate AIA:**

Select this option to specify to follow any URIs specified in the certificate’s Authority Information Access (AIA) extension for path discovery. The AIA extension specifies where to find up-to-date certificates. When this option is deselected, no URIs are processed in the AIA extension from the certificate. By default, the option is deselected.

**Basic Constraints Extension Required in CA Certificates:**

Select this option to specify that the certificate authority (CA) Basic Constraints certificate extension must be present for CA certificates. Some early German certified root certificates (7 and earlier) are not compliant to RFC3280

and do not contain the basic constraint extension. If it is known that a user's EE certificate chains up to such a German root, deselect this option. When this option is deselected, the presence of the CA Basic Constraints certificate in CA certificates is not required. By default, the value is selected.

**Require Valid Certificate Signature During Chain Building:**

Select this option to require that all Digital Signature Algorithm (DSA) signatures on certificates be valid before a chain is built. For example, in a chain CA > ICA > EE where the signature for EE is not valid, the chain building stops at ICA. EEs are not included in the chain. When this option is deselected, the entire chain is built regardless of whether an invalid DSA signature is encountered. By default, the option is deselected.

*OCSP Options Spec*

(Optional) An `OCSPOptionSpec` value that represents settings for using Online Certificate Status Protocol (OCSP) revocation checking. To provide a literal value, specify the following options.

**URL to Consult Option:** Sets the list and order of the OCSP servers used to perform the revocation check. Select one of these values:

**UseAIAInCert:**

(Default) Use the URL of an online certificate status protocol server specified in the Authority Information Access (AIA) extension in the certificate. The AIA extension is used to identify how to access certificate authority (CA) information and services for the issuer of the certificate.

**LocalURL:**

Use the specified URL for the OCSP server specified in the OCSP Server URL option.

**UseAIAIfPresentElseLocal:**

Use the URL of the OCSP server specified in the AIA extension in the certificate if present. If the AIA extension is not present in the certificate, use the URL that is configured in the OCSP Server URL.

**UseAIAInSignerCert:**

Use the URL of the OCSP server specified in the AIA extension in the OCSP request of the signer certificate.

**OCSP Server URL:**

Sets the URL of the configured OCSP server. The value is used only when the LocalURL or UseAIAIfPresentElseLocal values are in URL To Consult Option.

**Revocation Check Style:**

Sets the revocation-checking style that is used for verifying the trust status of the CRL provider's certificate from its observed revocation status. Select one of these values:

Sets the URL of the configured OCSP server. The value is used only when the LocalURL or UseAIAIfPresentElseLocal values are in URL To Consult Option.

Revocation Check Style:

Sets the revocation-checking style that is used for verifying the trust status of the CRL provider's certificate from its observed revocation status. Select one of these values:

**NoCheck:**

Does not check for revocation.

**BestEffort:**

Checks for revocation of all certificates when possible.

**CheckIfAvailable:**

(Default) Checks for revocation of all certificates only when revocation information is available.

**AlwaysCheck:**

Checks for revocation of all certificates.

**Max Clock Skew Time (Minutes):**

Sets the maximum allowed skew, in minutes, between response time and local time. Valid skew times are `0` - `2147483647` min. The default value is `5` min.

**Response Freshness Time (Minutes):**

Sets the maximum time, in minutes, for which a preconstructed OCSP response is considered valid. Valid response freshness times are `1`- `2147483647` min. The default value is `525600` min. (one year).

**Send Nonce:**

Select this option to send a nonce with the OCSP request. A *nonce* is a parameter that varies with time. These parameters can be a timestamp, a visit counter on a web page, or a special marker. The parameter is intended to limit or prevent the unauthorized replay or reproduction of a file. When the option deselected, a nonce is not sent with the request. By default, the option is selected.

**Sign OCSP Request:**

Select this option to specify that the OCSP request must be signed. When the option is deselected, the OCSP request does not need be signed. By default, the option deselected.

**Request Signer Credential Alias:**

Sets the credential alias used for signing the OCSP request when signing is enabled.

**Go Online for OCSP:**

Select this option to access the network for OCSP information. The network can be accessed to retrieve OCSP information for OCSP checking. The AEM Forms Server uses embedded and cached OCSP information when possible to reduce the amount of network traffic generated due to OCSP checking. When the option is deselected, OCSP checking is not retrieved from the network, and only embedded and cached OCSP information is used. By default, the option is selected.

**Ignore Validity Dates:**

Select this option to use the OCSP response thisUpdate and nextUpdate times. Ignoring these response times prevents any negative effect on response validity. The thisUpdate and nextUpdate

times are retrieved from external sources by using HTTP or LDAP, and can be different for each revocation information. When the option is deselected, the thisUpdate and nextUpdate times are ignored. By default, the option is deselected.

**Allow OCSP NoCheck Extension:**

Select this option to allow an OCSPNoCheck extension in the response signing certificate. An OCSP-NoCheck extension can be present in the OCSP Responder's certificate to prevent infinite loops from occurring during the validation process. When the option is deselected, the OCSPNoCheck extension is not used. By default, the option is selected.

**Require OCSP ISIS-MTT CertHash Extension:**

Select this option to specify that certificate public key hash (CertHash) extensions must be present in OCSP responses. This extension is required for SigQ validation. SigQ compliance requires the CertHash extension to be in the OCSP responder certificate. Select this option when processing for SigQ compliance and supported OCSP responders. When the option is deselected, the CertHash extension presence in the OCSP response is not required. By default, the option is deselected.

Sets the maximum allowed skew, in minutes, between response time and local time. Valid skew times are `0` - `2147483647` min. The default value is `5` min.

Response Freshness Time (Minutes):

Sets the maximum time, in minutes, for which a preconstructed OCSP response is considered valid. Valid response freshness times are `1`- `2147483647` min. The default value is `525600` min. (one year).

Send Nonce:

Select this option to send a nonce with the OCSP request. A *nonce* is a parameter that varies with time. These parameters can be a timestamp, a visit counter on a web page, or a special marker. The parameter is intended to limit or prevent the unauthorized replay or reproduction of a file. When the option deselected, a nonce is not sent with the request. By default, the option is selected.

Sign OCSP Request:

Select this option to specify that the OCSP request must be signed. When the option is deselected, the OCSP request does not need be signed. By default, the option deselected.

Request Signer Credential Alias:

Sets the credential alias used for signing the OCSP request when signing is enabled.

Go Online for OCSP:

Select this option to access the network for OCSP information. The network can be accessed to retrieve OCSP information for OCSP checking. The AEM Forms Server uses embedded and cached OCSP information when possible to reduce the amount of network traffic generated due to OCSP checking. When the option is deselected, OCSP checking is not retrieved from the network, and only embedded and cached OCSP information is used. By default, the option is selected.

Ignore Validity Dates:

Select this option to use the OCSP response thisUpdate and nextUpdate times. Ignoring these response times prevents any negative effect on response validity. The thisUpdate and nextUpdate times are retrieved from external sources by using HTTP or LDAP, and can be different for each revocation information. When the option is deselected, the thisUpdate and nextUpdate times are ignored. By default, the option is deselected.

Allow OCSP NoCheck Extension:

Select this option to allow an OCSPNoCheck extension in the response signing certificate. An OCSP-NoCheck extension can be present in the OCSP Responder's certificate to prevent infinite loops from occurring during the validation process. When the option is deselected, the OCSPNoCheck extension is not used. By default, the option is selected.

Require OCSP ISIS-MTT CertHash Extension:

Select this option to specify that certificate public key hash (CertHash) extensions must be present in OCSP responses. This extension is required for SigQ validation. SigQ compliance requires the CertHash extension to be in the OCSP responder certificate. Select this option when processing for SigQ compliance and supported OCSP responders. When the option is deselected, the CertHash extension presence in the OCSP response is not required. By default, the option is deselected.

*CRL Options Spec*

(Optional) A `CRLOptionSpec` value that represents the certificate revocation list (CRL) preferences when CRL is used to perform revocation checking. If you provide a literal value, specify the following options.

**Consult Local URI First:**

Select this option to use the CRL location provided as a local URI before any specified locations within a certificate. The CRL location provided is used for revocation checking. When this option is selected, it means the local URI is used first. When this option is deselected, the locations specified in the certificate before using the local URI are used. By default, the option is deselected.

**Local URI for CRL Lookup:**

Sets the URL for the local CRL store. This value is used only if the Consult Local URI First option is selected. No default value is provided.

**Revocation Check Style:**

Sets the revocation-checking style used for verifying the trust status of the CRL provider's certificate from its observed revocation status. Select one of these values:

- **NoCheck:** Does not check for revocation.
- **BestEffort:** (Default) Checks for revocation of all certificates when possible.
- **CheckIfAvailable:** Checks for revocation of all certificates only when revocation information is available.
- **AlwaysCheck:** Checks for revocation of all certificates.

**LDAP Server:**

Sets the URL or path of the Lightweight Directory Access Protocol (LDAP) server used to retrieve information about the certificate revocation list (CRL). The LDAP server searches for CRL information using the distinguished name (DN) according to the rules specified in RFC 3280

, section 4.2.1.14. For example, you can type `www.ldap.com` for the URL or `ldap://ssl.ldap.com:200` for the path and port. No default value is provided.

**Go Online for CRL Retrieval:**

Select this option to access the network to retrieve CRL information. CRL information is cached on the server to improve network performance. CRL information is retrieved online only when necessary. When this option is deselected, CRL information is not retrieved online. By default, the option is selected.

**Ignore Validity Dates:**

Select this option to use thisUpdate and nextUpdate times. Ignoring the response's thisUpdate and nextUpdate times prevents any negative effect on response validity. The thisUpdate and nextUpdate times are retrieved from external sources by using HTTP or LDAP and can be different for each revocation information. When the option is deselected, the thisUpdate and nextUpdate time are ignored. By default, the option deselected.

**Require AKI Extension in CRL:**

Select this option to specify that the Authority Key Identifier (AKI) extension must be present in the CRL. The AKI extension can be used for CRL validation. When this option is deselected, the presence of the AKI extension the CRL is not required. By default, the option is deselected.

*TSP Options Spec*

(Optional) A `TSPOptionSpec` value that represents the settings that define timestamp information applied to the certified signature.

If you provide a literal value, specify the following options.

**Time Stamp Server URL:**

Sets the URL for a TSP server. If no value is provided, the timestamp from the local system is applied. No default value is provided.

**Time Stamp Server Username:**

Sets the user name, if necessary, for accessing the TSP server. No default value is provided.

**Time Stamp Server Password:**

Sets the password for the user name, if necessary, for accessing the TSP server. No default value is provided.

**Time Stamp Server Hash Algorithm:**

Sets the hash algorithm used to digest the request sent to the timestamp provider. Select one of these values:

**SHA1: (Default)**

The Secure Hash Algorithm that has a 160-bit hash value.

**SHA256:**

The Secure Hash Algorithm that has a 256-bit hash value.

**SHA384:**

The Secure Hash Algorithm that has a 384-bit hash value.

**SHA512:**

The Secure Hash Algorithm that has a 512-bit hash value.

**RIPEMD160:**

The RACE Integrity Primitives Evaluation Message Digest that has a 160-bit message digest algorithm and is not FIPS-compliant.

**Revocation Check Style:**

Sets the revocation-checking style used for verifying the trust status of the CRL provider's certificate from its observed revocation status. Select one of these values:

**NoCheck:**

Does not check for revocation.

**BestEffort:**

(Default) Checks for revocation of all certificates when possible.

**CheckIfAvailable:**

Checks for revocation of all certificates only when revocation information is available.

**AlwaysCheck:**

Checks for revocation of all certificates.

**Use Expired Timestamps:**

Select this option to use timestamps that have expired during the validation of the certificate. When this option is deselected, expired timestamps are not used. By default, this option is selected.

**Predicted Time Stamp Token Size (In Bytes):**

Sets the estimated size, in bytes, of the TSP response. The size is used to create a signature hole in the PDF document. This value represents the maximum size of the timestamp response that the configured TSP could return. Configuring an undersized value can cause the operation to fail;

however, configuring an oversized value causes the size to be larger than necessary. It is recommended that this value is not modified unless the timestamp server requires a response size to be less than 4096 bytes. Valid values are from `60` to `10240`. The default value is `4096`.

**Send Nonce:**

Select this option to send a nonce with the request. A *nonce* is a parameter that varies with time. These parameters can be a timestamp, a visit counter on a web page, or a special marker. The parameter is intended to limit or prevent the unauthorized replay or reproduction of a file. When the option deselected, a nonce is not sent with the request. By default, the option is selected.

*Properties Map To Be Passed To SPI*

(Optional) A `java.utils.properties` value that specifies the properties passed to the service provider interface (SPI) to verify the certificate.

If you provide a literal value, clicking the ellipsis button opens the Select Asset dialog box. (See AboutSelectAsset.)

The file contains a property-value pair formatted as *[property name]=[value]*, where *[property name]* is the name of the property and *[value]* is the value assigned to the property.

**Output properties**

Property to specify the verification result.

*PDF Signature Verification Result*

The location in the process data model to store the operation verification results. The verification results include information about the signature and its validity status. The data type is `PDFSignatureVerificationResult`.

**Exceptions**

This operation can throw PDFOperationException, InvalidArgumentException, MissingSignatureFieldException, SignatureFieldNotSignedException, SignatureVerifyException, and PermissionsException exceptions.

## Verify XML Signature operation

Verifies an XML signature associated with a signature field. After the verification occurs, an error occurs to indicate that the document was signed in the future. The error occurs because the signature, which is being verified, was applied at a time that is more than 65 min. You can change this setting in administration console. (See Applications andServicesAdministrationHelp

).

For example, your application requires that XML signatures are validated on PDF forms. An XML signature signs the entire XDP package for PDF forms created in Designer. You use the XML Signature operation to validate that the signature validity of the signature and the identity and trust chain of the signer.

For information about the General and Route Evaluation property groups, see Commonoperationproperties.

**Common properties**

Properties to specify the input XML document and various verification information.

*Input XML*

A `document` value that represents an XML document that contains an XML signature to verify.

If you provide a literal value, clicking the ellipsis button opens the Select Asset dialog box. (See AboutSelectAsset.)

When you provide a PDF document that has signature fields, it populates the Signature Field Name property as a list. The list contains a list of fully qualified names of signature fields in the PDF document.

*Signature Field Name*

A `string` value that represents the value of the ID attribute of the signature node in the signed XML.

*Revocation Check Style*

A `RevocationCheckStyle` value that specifies the revocation-checking style used for verifying the trust status of the CRL provider's certificate from its observed revocation status.

If you provide a literal value, select one of these values:

**NoCheck:**

Does not check for revocation.

**BestEffort:**

Checks for revocation of all certificates when possible.

**CheckIfAvailable:**

(Default) Checks for revocation of all certificates only when revocation information is available.

**AlwaysCheck:**

Checks for revocation of all certificates.

*Verification Time*

(Optional) A `VerificationTime` value that specifies the verification time to use. The default value is Secure Time Else Current Time.

If you provide a literal value, select one of these values:

**Signing Time:**

The time that the signature was applied as given by the signer's computer.

**Current Time:**

The time that the verification operation is being carried out.

**Secure Time Else Current Time:**

The time specified by a trusted time-stamping authority.

If you specify Secure Time Else Current Time and validation returns a status of unknown with a trusted timestamp, the validation is checked using Current Time.

**Advanced properties**

Properties to specify path validation and other advanced settings.

*Path Validation Options Spec*

(Optional) A `PathValidationOptionSpec` that represents the settings that control RFC3280-related path validation options. For example, you can indicate whether policy mapping is allowed in the certification path. (See RFC3280

). If you provide a literal value, you can set the following options.

**Require Explicit Policy:**

Select this option to specify that the path must be valid for at least one of the certificate policies in the user initial policy set. When this option is deselected, the path validity is not required. By default, the option is deselected.

**Inhibit Any Policy:**

Select this option to specify that a policy object identifier (OID) must be processed if it is included in a certificate. When deselected, any policy can be selected. By default, the option is deselected.

**Check All Paths:**

Select this option to require that all paths to a trust anchor must be validated. When this option is deselected, all paths to a trust anchor are not validated. By default, the option is deselected.

**Inhibit Policy Mapping:**

Select this option to allow policy mapping in the certification path. When this option is deselect, policy mapping is not allowed in the certification path. By default, the option is deselected.

**LDAP Server:**

Sets the URL or path of the Lightweight Directory Access Protocol (LDAP) server used to retrieve information about the certificate revocation list (CRL). The LDAP server searches for CRL information using the distinguished name (DN) according to the rules specified in RFC 3280

, section 4.2.1.14. For example, you can type `www.ldap.com` for the URL or `ldap://ssl.ldap.com:200` for the path and port. No default value is provided.

**Follow URIs in Certificate AIA:**

Select this option to specify to follow any URIs specified in the certificate's Authority Information Access (AIA) extension for path discovery. The AIA extension specifies where to find up-to-date certificates. When this option is deselected, no URIs are processed in the AIA extension from the certificate. By default, the option is deselected.

**Basic Constraints Extension Required in CA Certificates:**

Select this option to specify that the certificate authority (CA) Basic Constraints certificate extension must be present for CA certificates. Some early German certified root certificates (7 and earlier) are not compliant to RFC3280

and do not contain the basic constraint extension. If it is known that a user's EE certificate chains up to such a German root, deselect this option. When this option is deselected, the presence of the CA Basic Constraints certificate in CA certificates is not required. By default, the value is selected.

**Require Valid Certificate Signature During Chain Building:**

Select this option to require that all Digital Signature Algorithm (DSA) signatures on certificates be valid before a chain is built. For example, in a chain CA > ICA > EE where the signature for EE is not valid, the chain building stops at ICA. EEs are not included in the chain. When this option is deselected, the entire chain is built regardless of whether an invalid DSA signature is encountered. By default, the option is deselected.

*OCSP Options Spec*

(Optional) An `OCSPOptionSpec` value that represents settings for using Online Certificate Status Protocol (OCSP) revocation checking. To provide a literal value, specify the following options.

**URL to Consult Option:** Sets the list and order of the OCSP servers used to perform the revocation check. Select one of these values:

**UseAIAInCert:**

(Default) Use the URL of an online certificate status protocol server specified in the Authority Information Access (AIA) extension in the certificate. The AIA extension is used to identify how to access certificate authority (CA) information and services for the issuer of the certificate.

**LocalURL:**

Use the specified URL for the OCSP server specified in the OCSP Server URL option.

**UseAIAIfPresentElseLocal:**

Use the URL of the OCSP server specified in the AIA extension in the certificate if present. If the AIA extension is not present in the certificate, use the URL that is configured in the OCSP Server URL.

**UseAIAInSignerCert:**

Use the URL of the OCSP server specified in the AIA extension in the OCSP request of the signer certificate.

**OCSP Server URL:**

Sets the URL of the configured OCSP server. The value is used only when the LocalURL or UseAIAIf-PresentElseLocal values are in URL To Consult Option.

**Revocation Check Style:**

Sets the revocation-checking style that is used for verifying the trust status of the CRL provider's certificate from its observed revocation status. Select one of these values:

Sets the URL of the configured OCSP server. The value is used only when the LocalURL or UseAIAIf-PresentElseLocal values are in URL To Consult Option.

Revocation Check Style:

Sets the revocation-checking style that is used for verifying the trust status of the CRL provider's certificate from its observed revocation status. Select one of these values:

**NoCheck:**

Does not check for revocation.

**BestEffort:**

Checks for revocation of all certificates when possible.

**CheckIfAvailable:**

(Default) Checks for revocation of all certificates only when revocation information is available.

**AlwaysCheck:**

Checks for revocation of all certificates.

**Max Clock Skew Time (Minutes):**

Sets the maximum allowed skew, in minutes, between response time and local time. Valid skew times are `0` - `2147483647` min. The default value is `5` min.

**Response Freshness Time (Minutes):**

Sets the maximum time, in minutes, for which a preconstructed OCSP response is considered valid. Valid response freshness times are `1`- `2147483647` min. The default value is `525600` min. (one year).

**Send Nonce:**

Select this option to send a nonce with the OCSP request. A *nonce* is a parameter that varies with time. These parameters can be a timestamp, a visit counter on a web page, or a special marker. The parameter is intended to limit or prevent the unauthorized replay or reproduction of a file. When the option deselected, a nonce is not sent with the request. By default, the option is selected.

**Sign OCSP Request:**

Select this option to specify that the OCSP request must be signed. When the option is deselected, the OCSP request does not need be signed. By default, the option deselected.

**Request Signer Credential Alias:**

Sets the credential alias used for signing the OCSP request when signing is enabled.

**Go Online for OCSP:**

Select this option to access the network for OCSP information. The network can be accessed to retrieve OCSP information for OCSP checking. The AEM Forms Server uses embedded and cached OCSP information when possible to reduce the amount of network traffic generated due to OCSP checking. When the option is deselected, OCSP checking is not retrieved from the network, and only embedded and cached OCSP information is used. By default, the option is selected.

**Ignore Validity Dates:**

Select this option to use the OCSP response thisUpdate and nextUpdate times. Ignoring these response times prevents any negative effect on response validity. The thisUpdate and nextUpdate times are retrieved from external sources by using HTTP or LDAP, and can be different for each revocation information. When the option is deselected, the thisUpdate and nextUpdate times are ignored. By default, the option is deselected.

**Allow OCSP NoCheck Extension:**

Select this option to allow an OCSPNoCheck extension in the response signing certificate. An OCSP-NoCheck extension can be present in the OCSP Responder's certificate to prevent infinite loops from occurring during the validation process. When the option is deselected, the OCSPNoCheck extension is not used. By default, the option is selected.

**Require OCSP ISIS-MTT CertHash Extension:**

Select this option to specify that certificate public key hash (CertHash) extensions must be present in OCSP responses. This extension is required for SigQ validation. SigQ compliance requires the CertHash extension to be in the OCSP responder certificate. Select this option when processing for SigQ compliance and supported OCSP responders. When the option is deselected, the CertHash extension presence in the OCSP response is not required. By default, the option is deselected.

Sets the maximum allowed skew, in minutes, between response time and local time. Valid skew times are `0` - `2147483647` min. The default value is `5` min.

Response Freshness Time (Minutes):

Sets the maximum time, in minutes, for which a preconstructed OCSP response is considered valid. Valid response freshness times are `1`- `2147483647` min. The default value is `525600` min. (one year).

Send Nonce:

Select this option to send a nonce with the OCSP request. A *nonce* is a parameter that varies with time. These parameters can be a timestamp, a visit counter on a web page, or a special marker. The

parameter is intended to limit or prevent the unauthorized replay or reproduction of a file. When the option deselected, a nonce is not sent with the request. By default, the option is selected.

Sign OCSP Request:

Select this option to specify that the OCSP request must be signed. When the option is deselected, the OCSP request does not need be signed. By default, the option deselected.

Request Signer Credential Alias:

Sets the credential alias used for signing the OCSP request when signing is enabled.

Go Online for OCSP:

Select this option to access the network for OCSP information. The network can be accessed to retrieve OCSP information for OCSP checking. The AEM Forms Server uses embedded and cached OCSP information when possible to reduce the amount of network traffic generated due to OCSP checking. When the option is deselected, OCSP checking is not retrieved from the network, and only embedded and cached OCSP information is used. By default, the option is selected.

Ignore Validity Dates:

Select this option to use the OCSP response thisUpdate and nextUpdate times. Ignoring these response times prevents any negative effect on response validity. The thisUpdate and nextUpdate times are retrieved from external sources by using HTTP or LDAP, and can be different for each revocation information. When the option is deselected, the thisUpdate and nextUpdate times are ignored. By default, the option is deselected.

Allow OCSP NoCheck Extension:

Select this option to allow an OCSPNoCheck extension in the response signing certificate. An OCSP-NoCheck extension can be present in the OCSP Responder's certificate to prevent infinite loops from occurring during the validation process. When the option is deselected, the OCSPNoCheck extension is not used. By default, the option is selected.

Require OCSP ISIS-MTT CertHash Extension:

Select this option to specify that certificate public key hash (CertHash) extensions must be present in OCSP responses. This extension is required for SigQ validation. SigQ compliance requires the CertHash extension to be in the OCSP responder certificate. Select this option when processing for SigQ compliance and supported OCSP responders. When the option is deselected, the CertHash extension presence in the OCSP response is not required. By default, the option is deselected.

*CRL Options Spec*

(Optional) A `CRLOptionSpec` value that represents the certificate revocation list (CRL) preferences when CRL is used to perform revocation checking. If you provide a literal value, specify the following options.

**Consult Local URI First:**

Select this option to use the CRL location provided as a local URI before any specified locations within a certificate. The CRL location provided is used for revocation checking. When this option is selected, it means the local URI is used first. When this option is deselected, the locations specified in the certificate before using the local URI are used. By default, the option is deselected.

**Local URI for CRL Lookup:**

Sets the URL for the local CRL store. This value is used only if the Consult Local URI First option is selected. No default value is provided.

**Revocation Check Style:**

Sets the revocation-checking style used for verifying the trust status of the CRL provider's certificate from its observed revocation status. Select one of these values:

- **NoCheck:** Does not check for revocation.
- **BestEffort:** (Default) Checks for revocation of all certificates when possible.
- **CheckIfAvailable:** Checks for revocation of all certificates only when revocation information is available.
- **AlwaysCheck:** Checks for revocation of all certificates.

**LDAP Server:**

Sets the URL or path of the Lightweight Directory Access Protocol (LDAP) server used to retrieve information about the certificate revocation list (CRL). The LDAP server searches for CRL information using the distinguished name (DN) according to the rules specified in RFC 3280

, section 4.2.1.14. For example, you can type `www.ldap.com` for the URL or `ldap://ssl.ldap.com:200` for the path and port. No default value is provided.

**Go Online for CRL Retrieval:**

Select this option to access the network to retrieve CRL information. CRL information is cached on the server to improve network performance. CRL information is retrieved online only when necessary. When this option is deselected, CRL information is not retrieved online. By default, the option is selected.

**Ignore Validity Dates:**

Select this option to use thisUpdate and nextUpdate times. Ignoring the response's thisUpdate and nextUpdate times prevents any negative effect on response validity. The thisUpdate and nextUpdate times are retrieved from external sources by using HTTP or LDAP and can be different for each revocation information. When the option is deselected, the thisUpdate and nextUpdate time are ignored. By default, the option deselected.

**Require AKI Extension in CRL:**

Select this option to specify that the Authority Key Identifier (AKI) extension must be present in the CRL. The AKI extension can be used for CRL validation. When this option is deselected, the presence of the AKI extension the CRL is not required. By default, the option is deselected.

### Output properties

Property to specify the verification result.

*XML Signature Verification Result*

The location in the process data model to store the information about the signature and its validity status. The data type is `XMLSignatureVerificationResult`.

### Exceptions

This operation can throw SignatureVerifyException, InvalidArgumentException, and MissingSignature-FieldException exceptions.

## Signature exceptions

The Signature service provides the following exceptions for throwing exception events.

### CredentialLoginException

Thrown if a problem occurs when a credential is not found or the credential expired. The credential is required to sign or certify.

### DuplicateSignatureFieldException

Thrown if a signature field is duplicated.

### FIPSComplianceException

Thrown when FIPS compliance is not met. When FIPS is required, an FIPSComplianceException error is thrown when MD5 or RIPEMD160 is used as a digest hashing algorithm.

### InvalidArgumentException

Thrown if an invalid argument that is required to perform a Signature operation is specified.

### MissingSignatureFieldException

Thrown if the specified signature field cannot be located within a PDF document.

### NoCertifyingSignatureException

Thrown if the input PDF has no certifying signature and the operation requires one to be present.

### PDFOperationException

Thrown if a specific operation cannot be performed on the input PDF document due to PDF-related processing errors.

### PermissionsException

Thrown if an operation is requested but the input PDF document has insufficient permissions to complete it.

### SeedValueValidationException

Thrown at the time of signing or certifying if the seed values associated with the signature field cannot be validated.

### SignatureFieldNotSignedException

Thrown if an attempt to sign a signature field failed.

### SignatureFieldSignedException

Thrown if the specified signature field already contains a signature.

### SignatureVerifyException

Thrown if a signature cannot be verified.

### SigningException

Thrown if an error occurs while digitally signing a PDF document.

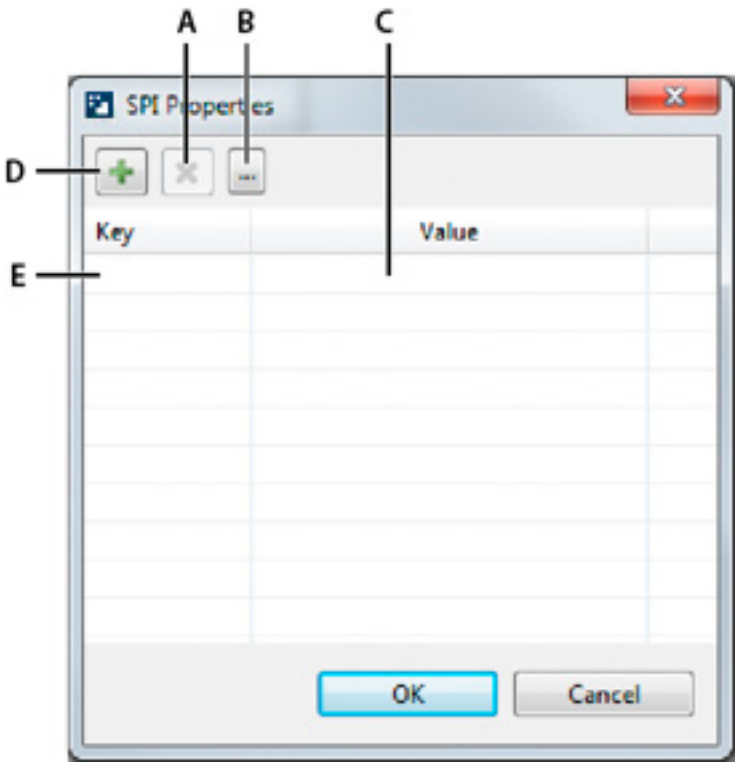
## SPI Properties

If you provide a literal value, clicking the ellipsis button opens the SPI Properties dialog box. Clicking the ellipsis button in the SPI Properties dialog box displays the Open dialog box. In the Open dialog box, you can select a file from your computer or from a network location. During run time, if you selected a file from a location on your computer, it must exist in the same location on the AEM Forms Server.

You can select properties in a file that has the following format:

```
key1=value1
key2=value2
key3=value3
```

**NOTE:** The keys and values that you use are specific to the SPI that you are connecting to.




**A.**

Removes the selected key and value pair.

**B.**

Opens the Open dialog box to load SPI properties from a file.

**C.**

Specifies the value assigned to the key.

**D.**

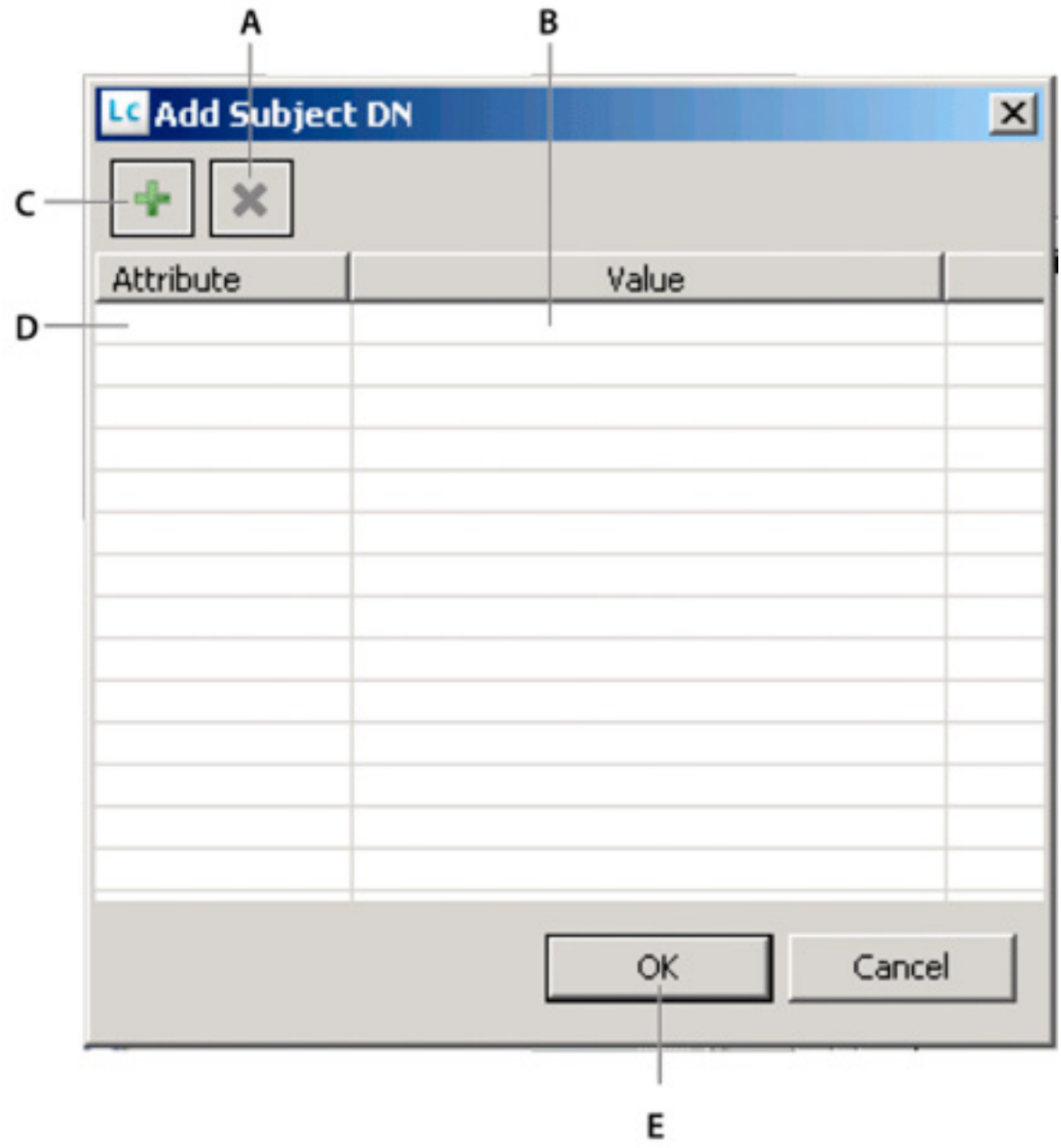
Adds new key and value pair.

**E.**

Specifies a new key.

## Add Subject DN

Use the Add Subject DN dialog box to add subject distinguished names to the Subject Distinguished Name option.




**A.**

Removes the selected value.

**B.**

Specifies the value assigned to the attribute.

**C.**

Adds a new entry for specifying an attribute and value.

**D.**

Specifies a new attribute name.

**E.**

Adds new subject distinguished names to list.

# 22.46. Stall

The Stall service provides the execute operation, which is useful for preventing situational errors that you anticipate may occur.

For information about using the Stall service, see Services Reference for AEM Forms

.

## execute operation

When used in asynchronous and synchronous branches, the execute operation stalls the branch. If you are aware of a possible situational error in your process, you can add a route that leads to this execute operation. You add a condition to the route that checks if the situational error has occurred. When the situation occurs, the branch is stalled so that you can fix the error and restart the process using the administration console.

**NOTE:** When used in transactional branches or short-lived processes, the execute operation throws an exception instead of stalling the branch.

For example, processes can use data that is provided from an external resource, such as a partner's database. A Script action can be used to verify that the data is valid. If the data is not valid, a Stall action can be executed that stalls the process instance while the data in the database is corrected.

For information about the General and Route Evaluation property groups, see Commonoperation properties.

### Input properties

Properties for configuring the stall message.

*Message*

A `string` value that provides a message that describes the reason that the branch is stalled. The message appears in the details about the stalled branch in the forms workflow pages of administration console.

# 22.47. Submit Form Guide

Lets you handle the submission of a form guide from within Workspace. You can use the Submit Form Guide service in a process map or to configure the submit service, which is a post-processing step for an xfaForm variable.

It is not recommended that you modify this service directly as it is possible to do so with Workbench. It is recommended that you make a copy of the service, modify it, and then use your copy as a post-processing step for an xfaForm variable.

**IMPORTANT:** This process may be updated during upgrades of AEM Forms and therefore, you may need to update your version of the service.

## invoke

Submits the form guide to start a process.

For information about the General and Route Evaluation property groups, see Commonoperation properties.

### Input

*Environment Buffer*

A `string` value that represents the HTTP header information, delimited by an ampersand (&), that includes the server name, content type, and HTTP reference. The user's environment buffer may appear like the following value:

```
CONTENT_TYPE=application/pdf&CONTENT_TYPE=application/vnd.adobe.xdp+xml&
CONTENT_TYPE=text/xml
```

*Submitted Data*

A `document` value that represents the data to merge with the form during rendering.

If you provide a literal value, when you click the ellipsis button ..., the Open dialog box appears, where the file that represents the data can be selected from the local computer or a network location.

*Target Url*

A `string` value that specifies the URL of the web application or servlet that the submitted HTML form is sent to.

*User Agent*

A `string` value that specifies a list of user's web browser settings, delimited by a semicolon (;). For example, such a list may appear like the following value:

```
Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1; .NET CLR
2.0.50727;
```

**Output**

*New Render Flag*

A `boolean` value that specifies whether the form must be rendered again to display to the user. A value of `True` specifies that the form guide must be displayed to the user because either an error occurred while trying to submit the form guide or form data was passed but not necessarily to submit the form. A value of `False` specifies that the form does not need to be displayed to the user.

*Output Document*

A `document` value that stores the form data.

*Submit Success Flag*

A `boolean` value that specifies whether the user submitted data. A value of `True` specifies that data was submitted and a value of `False` specifies that data was not submitted.

**Invocation Policy**

Specifies whether to wait for a response after invoking the operation by selecting either Do Not Wait For Response or Wait For Response. The default is `Wait For Response`.

*Do Not Wait For Response*

Specifies that no response is required from the server.

*Wait For Response*

Specifies that a response is required from the server before further processing.

# 22.48. Submit HTML Form

Lets you handle the submission of an HTML form from within Workspace. You can use this service in a process map or to configure the submit service, which is a post-processing step for an xfaForm variable.

It is not recommended that you modify this service directly as it is possible to do so with Workbench. It is recommended that you make a copy of the service, modify it, and then use your copy as a post-processing step for an xfaForm variable.

**IMPORTANT:** This process may be updated during upgrades of AEM Forms and therefore, you may need to update your version of the service.

## invoke operation

Submits an HTML form to start a process.

For information about the General and Route Evaluation property groups, see Commonoperation properties.

### Input properties

*Environment Buffer*

A `string` value that represents the HTTP header information, delimited by an ampersand (&), that includes the server name, content type, and HTTP reference. The user's environment buffer may appear like the following value:

```
CONTENT_TYPE=application/pdf&CONTENT_TYPE=application/vnd.adobe.xdp+xml&
CONTENT_TYPE=text/xml
```

*Submitted Data*

A `document` value that represents the data to merge with the form during rendering.

If you provide a literal value, when you click the ellipsis button ..., the Open dialog box appears where the file that represents the data can be selected from the local computer or a network location.

*Target Url*

A `string` value that specifies the URL of the web application or servlet that the submitted form data is sent to.

*User Agent*

A `string` value that specifies a list of the user's web browser settings, delimited by a semicolon (;). For example, such a list may appear like the following value:

```
Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1; .NET CLR
2.0.50727;
```

### Output properties

*New Render Flag*

A `boolean` value that specifies whether the form needs to rendered again to display to the user. A value of `True` specifies that the HTML form needs to be displayed to the user because either an error occurred while trying to submit the HTML form or form data was passed but not necessarily to submit the form. A value of `False` specifies that the form does not need to be displayed to the user.

*Output Document*

A `document` value that stores the form data.

*Selected Route*

A `string` value that stores the name of the route the user selected from within Workspace.

*Submit Success Flag*

A `boolean` value that specifies whether the user submitted data. A value of `True` specifies that data was submitted and a value of `False` specified that data was not submitted.

**Invocation Policy properties**

Specifies whether to wait for a response after invoking the operation by selecting either Do Not Wait For Response or Wait For Response. The default is `Wait For Response`.

*Do Not Wait For Response*

Specifies that no response is required from the server.

*Wait For Response*

Specifies that a response is required from the server before further processing.

# 22.49. Submit PDF Form

Lets you handle the submission of a PDF form from within Workspace. You can use this service in a process map or to configure the submit service, which is a post-processing step for an xfaForm variable.

It is not recommended that you modify this service directly as it is possible to do so with Workbench. It is recommended that you make a copy of the service, modify it, and then use your copy as a post-processing step for an xfaForm variable.

**IMPORTANT:** This process may be updated during upgrades of AEM Forms and therefore, you may need to update your version of the service.

## invoke operation

Submits a PDF Form to start a process.

For information about the General and Route Evaluation property groups, see Commonoperation properties.

**Input properties**

*Environment Buffer*

A `string` value that represents the HTTP header information delimited by an ampersand(&) that includes the server name, content type, and HTTP reference. The user's environment buffer may appear like the following value:

```
CONTENT_TYPE=application/pdf&CONTENT_TYPE=application/vnd.adobe.xdp+xml&
CONTENT_TYPE=text/xml
```

*Submitted Data*

A `document` value that represents the data to merge with the form during rendering.

If you provide a literal value, when you click the ellipsis button ..., the Open dialog box appears where the PDF file representing the data can be selected from the local computer or a network location.

*Target Url*

A `string` value that specifies the URL of the web application or servlet that the submitted form data is sent to.

*User Agent*

A `string` value that specifies a list of the user's web browser settings, delimited by a semicolon (;). For example, such a list may appear like the following value:

```
Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1; .NET CLR
2.0.50727;
```

**Output properties**

*Error Document*

A `document` value that stores any validation errors as UTF8-encoded XML. For more information about the format of the XML encoding, see FormsResult.

*New Render Flag*

A `boolean` value that specifies whether the form must be rendered again to display to the user. A value of `True` specifies that the PDF form must be displayed to the user because either an error occurred while trying to submit the PDF form or form data was passed but not necessarily to submit the form. A value of `False` specifies that the form does not need to be displayed to the user.

*Output Document*

A `document` value that stores the form data.

*Selected Route*

A `string` value that stores the name of the route the user selected from within Workspace.

*Submit Success Flag*

A `boolean` value that specifies whether the user submitted data. A value of `True` specifies that data was submitted and a value of `False` specifies that data was not submitted.

**Invocation Policy properties**

Specifies whether to wait for a response after invoking the operation by selecting either Do Not Wait For Response or Wait For Response. The default is `Wait For Response`.

*Do Not Wait For Response*

Specifies that no response is required from the server.

*Wait For Response*

Specifies that a response is required from the server before further processing.

# 22.50. User 2.0

Lets you involve people in your processes.

For information about using the User service to involve people in processes, see Designing human-centric processes and Services Reference AEM Forms

.

## Assign Multiple Tasks operation

This operation is used in the design of human-centric processes. It creates multiple task and assigns them to users and groups. For information about assigning tasks, see Assigning tasks to users.

### Participants properties

Properties for specifying who is assigned the tasks that are generated. You can specify a user list or specific users and groups. You can search for users and groups to express literal values, or use variables and XPath expressions.

You can add as many participants as needed. After you add them, they appear in the table. The following buttons enable you to add and remove participants:

- Click to add a participant. After clicking, select how to specify the participants:
  - **User List:** Select a user list from an application or create a user list. (See Selecta user list.)
  - **User:** Search for a specific user to assign a task to. (See About Select User.)
  - **Group:** Search for a specific group to assign a task to. (See AboutSelect Group.)
  - **Variable:** Select a variable that identifies a user or group. The available variables appear in the Select Variable dialog box.
  - **XPath Expression:** Create an XPath expression that evaluates to the identifier of a user or group.

  *To read about the type of information that a variable or XPath expression must provide, see Usean XPath expression or variable to specify users.)*
- Click to remove the selected item from the table.

For each of the items that you add to the table, you can configure the following properties:

**Assign To Every User In The Group:**

For items that represent a user group, select this option to assign a task to every member of the group. Otherwise, the task is sent to the group's To Do list.

**Allow Out Of Office Designation:**

Select this option to forward the task based on the user's Workspace out-of-office settings.

### Task Instructions properties

Describes what the user must do to complete the task.

*Task Instructions*

A `string` value that represents the instructions for the task. If you provide a literal value, type the instructions for the task. For information about how to format the instructions using HTML, see Providingtask instructions and Task instructionson task cards.

**TIP:** Click the Insert Expression button to provide a template value to include the results of XPath expressions with literal text.

**User Actions properties**

Properties that determine actions that users can select when they complete tasks in workspace. You can configure actions so that when users click them, one or both of the following results occur:

- A specific route is followed. (See Providingactions for submitting tasks.)
- A confirmation message appears to the user. (See Requireconfirmation when submitting tasks.)

Actions appear in workspace in the order that they appear in this list. Use these buttons to create, modify, and order the actions.

Click to add an action. After clicking, specify property values in the Action Properties dialog box.

Click to modify the selected action using the Action Properties dialog box.

Click to remove the selected action.

Click to move the selected action up in the list.

Click to move the selected action down in the list.

*Use Completion Policies*

Select to use completion policies for completing the Assign Multiple Tasks operation before all tasks are completed. (See Addingcompletion policies to Assign Multiple Tasks operations.)

*Action Properties dialog box*

The Action Properties dialog box exposes the following properties:

**Action Name:**

The name of the action that appears in Workspace. ***NOTE:*** *Do not use commas in action names.*

**This Action Needs Confirmation, And Will Use The Following Text As The Message:**

Select to display a message box to the user when they click the action. In the box, type the message.

**TIP:** Click the ellipsis button to open XPath Builder and create an XPath expression to include in the message. At run time, the evaluated value of the expression is inserted in the message.

**Presentation & Data properties**

Properties for displaying an asset to users. For information about how to present assets and data to users, see Designingdata capture and presentation.

To specify the type of file to display, select one of the following options:

**Use An Application Asset:**

Select to display an application asset, such as a form or Guide, to the user.

**Use A Document Variable:**

Select to display the document that is stored in a document variable.

*Application Asset*

Properties for specifying the asset and data to display to users. These properties appear when Use An Application Asset is selected.

**Asset:**

The presentation asset to display to the user. Click the ellipsis button  and select an asset from any application.

**Action Profile:**

The action profile to use with the asset. Select the action profile from the list. The action profiles that appear are already created for the asset that you selected.

**Initial Task Data:**

The data to merge with the asset. Select an xml variable that contains the data. If the action profile that you selected includes a prepare data process, the variable is passed to that process before merging with the asset.

**Prepare Data Process Properties:**

The property group that appears when the prepare data process of the selected action profile includes input or output variables. Provide values for the properties that appear.

**Render Process Properties:**

The property group that appears when the render process of the selected action profile includes input or output variables. Provide values for the properties that appear.

**Submit Process Properties:**

The property group that appears when the submit process of the selected action profile includes input or output variables. Provide values for the properties that appear.

*Document Variable*

Property for specifying the document variable that holds the file to display to the user. This property group appears when Use A Document Variable is selected.

**Variable:**

Select the document variable to use.

**TIP:** If the variable that holds the file is not yet created, click the plus sign (+) button to create a variable. Later, configure your process so that the variable is populated with a value.

*Reader Submit*

Configure these properties if users are opening tasks using Adobe Reader.

**Submit Via Reader:**

Select if users open and submit tasks using Adobe Reader. When selected, a submit button is displayed in Workspace. If users open tasks using Acrobat Professional and Acrobat Standard, the PDF form includes one or more submit buttons. Workspace detects the buttons and displays them accordingly.

**Submit As:**

If Submit Via Reader is selected, specify the type of data that is submitted when users complete their tasks.

**Output properties**

Properties for storing the data that is sent to the AEM forms Server when users complete tasks. Saving the data is necessary if you want to use the data later in the process (for example, when evaluating the results of a document review). For more information, see Working with captured data.

**TIP:** For the following properties, if the variable required to store data is not yet created, click the plus sign (+) button to create a variable. If you are using an XPath expression, click the ellipsis button to open XPath Builder.

*Task Result Collection*

Properties for storing submitted Task Result values in a Task Result Collection value.

**Variable:**

The Task Result Collection variable in which to store the Task Result value that is submitted. If the variable is not yet created, click the plus sign button to create a variable.

**Include Captured Data:**

Select to include field data from tasks in the Task Result values that are stored in the collection. If you are interested in only task metadata, do not select this option. For example, do not select it if you are evaluating only the user actions that were clicked for each task.

**Include Attachments:**

Select to include task attachments in the Task Result values that are stored in the collection.

**Workspace User Interface properties**

Properties for specifying which task tools appear in Workspace when users open the task. If you configure these properties using a variable, it must be a Task Runtime UI variable. To provide literal values, specify values for the following properties.

**Default:**

Select to display the standard Workspace features.

**Approval Container (Deprecated):**

Select to display features for reviewing documents. The features enable users to see the status of the review, add comments, and see the task instructions..

**Custom:**

Select to display a custom set of tools. Custom tools are created by using the AEM Forms SDK. (See Programming with AEM Forms

.) Click Browse to specify where the executable files are located on the AEM Forms Server.

*UI Options*

Properties for specifying how the form is opened.

**Form Must Be Displayed When Completing:**

Select when users must open the task before completing it. If you do not select this option, users can complete the task from their To Do list without opening the form.

**Open Form Full-screen:**

Select to display the form using all the available space in the web browser window. The form obscures the Workspace UI when the user opens the task. Whether this option is selected or not, users can maximize or minimize the display area of the form as needed.

**Task Access Control List (ACL) properties**

Properties of the access control list (ACL) for tasks. ACLs control which Workspace features are exposed to users. For information, see Configuring access to task functionality.

The ACL includes a default entry, called *<default ACL>,*that applies to all users. Add a user to the list to override the default access control:

- Click Add to search for the user to add to the list.
- Click the ellipsis button to open XPath Builder and create an XPath expression that identifies a user. The expressions must evaluate to one of the following types of information:
  - Global unique identifier (GUID) of the user account, such as `9A7AD945-CA53-11D1-BBD0-0080C76670C0`
  - Login name of the user, such as `atanaka`
  - Canonical name of the user, such as `atanaka.sampleorganization.com`
  - Eemail address, such as `atanaka@sampleorganization.com`
  - Common name, such as `Akira Tanaka`
  - User value that represents the user.
- To remove a user from the list, select the user and click Delete. You cannot remove the default item.

You can control access to the following features:

**Claim:**

Claim a task. When users have this permission, they can claim a task from another user's queue.

**Add Notes:**

Add notes to the task. In addition, the user can set read, modify, and delete rights on each new note.

**Share:**

Share the task. When a task is shared, the original permissions are enforced, and the user with whom the task is shared can claim the task as their own.

**Forward:**

Delegate the task to another user.

**Add Attachments:**

Add attachments to the task. In addition, the user can set read, modify, and delete rights on each new attachment.

**Consult:**

Consult the task. Consulting is similar to forwarding the task, but the consultant cannot complete the task. The consultant can only open the task, save the task, add attachments and notes, and return the task to the user who requested the consult.

**Add ACL For Shared Queue:**

Add permissions for other users who have shared queue access to the assigned user's queue.

**Attachments properties**

Properties that specify whether attachments and notes are exposed in Workspace. Properties also control where the notes and attachments from the task are saved when the task is submitted.

**Show Attachment Window For This Task:**

Select to allow users to see task attachments and notes.

**Input List:**

(Optional) A List of document values that represent the attachments and notes to add to the task.

**Priority properties**

(Optional) A `Task Priority` value that specifies a priority of `Highest`, `High`, `Normal`, `Low`, or `Lowest`. The default is `Normal`. When specifying a literal value, select the priority from the list.

For more information, see Specifying task priority.

**Reminders properties**

Properties for configuring task reminders. For more information, see Sending reminders about tasks.

You can specify the following characteristics of reminders:

- When the first reminder is sent
- The duration between subsequent reminders
- The task instructions when the reminder occurs

To specify the value as a variable or XPath expression, a `TaskReminder` value must be provided. To provide a literal value, provide values for the following properties:

**Enable First Reminder:**

Select this option to send a reminder to the task owner.

**Use Business Calendar:**

- Select to use business days instead of calendar days to calculate reminder dates. When this option is selected, the Hours and Minutes boxes are unavailable, and the business calendar associated with the selected user is used. If the selected user does not have a business calendar configured, the default business calendar is used.
- **Days, Hours, and Mintues:**Specify the amount of time (in days, hours, and minutes) after the task is first assigned when the reminder occurs. If the Days, Hours, and Minutes boxes all have a value of `0`, the reminder does not occur.

**Enable Repeat Reminder:**

Select this option to send reminders at regular intervals after the first reminder is sent. The time when repeat reminders occur are calculated from the time that the first reminder is sent.

- **Use Business Calendar:** Select this option to use business days instead of calendar days to calculate repeated reminder dates. When this option is selected, the Hours and Minutes boxes are unavailable, and the business calendar associated with the selected user is used. If the selected user does not have a business calendar configured, the default business calendar is used.
- **Days, Hours, and Mintues:** Specify the amount of time (in days, hours, and minutes) after the task is first assigned when the reminder occurs. If the Days, Hours, and Minutes boxes all have a value of `0`, the reminder does not occur.

**Change Task Instructions On Reminder:**

Select this option to change the task instructions when reminders occur. Type the template for the instructions in the box. Click the ellipsis button to incorporate process data using XPath expressions. For information about templates, see .

**Deadline properties**

Properties for configuring task deadlines. You can specify when the deadline occurs, whether to change task instructions when the deadline occurs, and whether a specific route is followed.

To configure deadlines by using a variable or XPath expression, a `Task Deadline` value must be provided. To provide a literal value, specify values for the following properties:

**Enable First Deadline:**

Select this option to create a deadline for the task.

**Use Business Calendar:**

Select this option to use business days instead of calendar days to calculate deadline dates. When this option is selected, the Hours and Minutes boxes are unavailable, and the business calendar associated with the assigned user is used. If the user does not have a business calendar configured, the default business calendar is used.

**Days, Hours, and Mintues:**

Specify the amount of time (in days, hours, and minutes) after the task is first assigned when the deadline occurs. If the Days, Hours, and Minutes boxes all have a value of `0`, the deadline does not occur.

**Change Task Instructions On Reminder:**

Select this option to change the task instructions when reminders occur. Type the template for the instructions in the box. Click the ellipsis button to incorporate process data by using XPath expressions.

**Follow A Specific Route On Deadline:**

Select this option to determine the next operation to execute by specifying the route to follow when the deadline occurs.

**Select Route:**

Select the route to follow when Follow A Specific Route is selected. If no routes originate from the Assign Task operation, There Are No Outbound Routes appears in the list.

**Custom Email Templates properties**

Properties for configuring the text in emails that are sent when a task is assigned, a reminder occurs, or a deadline occurs. In the list, select the event and then configure the email template properties:

**Use Server Default:**

Use the email template configured on the server.

**Do Not Send Email:**

Do not send an email notification.

**Customize:**

Modify the email template for this task. Click Edit Email Template to use the Email Template Editor dialog box to modify the email template for the specified action. (See AboutEmail Template Editor.)

**Exceptions**

The Assign Multiple Tasks operation can throw an InvalidPrincipal exception. The exception occurs when a user or group that no longer exists is specified as a property value.

RELATED LINKS:

Specifying the Workspace ES2 user interface

Document review and approval processes

Specifying template expressions

About business calendars

## Assign Task operation

This operation is used in the design of human-centric processes. It creates a task and assigns it to a user or group. You can configure the following task characteristics (See Involvingusers in processes.):

- The behavior of the task at run time
- Which Workspace features people can use when they open the task in Workspace or when they use email to complete a task
- The content of email messages that are sent as task notifications to users

For information about the General and Route Evaluation property groups, see Commonoperation properties.

### Initial User Selection properties

Property to specify the user or group to assign the task to. For more information about assigning tasks, see Creatingtasks.

*Select Initial User*

The user or user group to assign the task.

**Allow Out Of Office Designation:**

Select this option to allow the task to be routed to another user when the assigned user is out of the office. The property is selected by default.

**Assign To Specific User:**

Select this option to specify a user to assign the task to. The user must exist in the AEM Forms environment. Use the Browse button to open the Select User dialog box and select the user profile. (See About Select User.)

**Assign To Process Initator:**

Select this option to specify that the user who started the process is assigned the task.

**Assign To Group:**

Select this option to assign the task to a user group. Users that belong to the group can be assigned a task in one of the following ways:

- **Assign To Group Queue:** Assigns the task to the group's shared To Do list.

- **Assign to Random User In Group:** Assigns the task to any user who belongs to the specified group profile.

  Use the Browse button to open the Select Group dialog box and select the group to assign the task. (See About Select Group.) ***NOTE:*** *Task assignments to a user or user group fail even though the user domains in the development and production environments are configured exactly the same. See Assigningtasks across development and production environments for more information.*

**XPath Expression:**

Select this option to create an XPath expression that evaluates to one of the following values:

- A `string` value that represents a user or group GUID, canonical name, login identification, email address, or common name. Use a common name or email address only if you are certain that they are unique.
- A `User` value.
- A `Group` value.

**TIP:** You can obtain User and Group values by using the User Lookup service. Use the User Lookup service in the process before the Assign Task operation. Save the results in a variable so that it is available in the XPath expression. (See UserLookup.*)*

**Task Instructions properties**

Describes what the user must do to complete the task.

*Task Instructions*

A `string` value that represents the instructions for the task. If you provide a literal value, type the instructions for the task. For information about how to format the instructions using HTML, see Providingtask instructions and Task instructionson task cards.

**TIP:** Provide a template value to include the results of XPath expressions with literal text.

**User Actions properties**

Properties that determine actions that users can select when they complete tasks in Workspace ES. You can configure actions so that when users click them, one or both of the following results occur:

- A specific route is followed. (See Providingactions for submitting tasks.)
- A confirmation message appears to the user. (See Requireconfirmation when submitting tasks.)

Actions appear in Workspace in the order that they appear in this list. Use these buttons to create, modify, and order the actions.

Click to add an action. After clicking, specify property values in the Action Properties dialog box.

Click to modify the selection action using the Action Properties dialog box.

Click to remove the selected action.

Click to move the selected action up in the list.

Click to move the selected action down in the list.

The Action Properties dialog box exposes the following properties:

**Action Name:**

The name of the action that appears in Workspace. ***NOTE:*** *Do no use commas with action names.*

**Destination:**

Select the operation that executes when the action is clicked.

**This Action Needs Confirmation, And Will Use The Following Text As The Message:**

Select to display a message box to the user when they click the action. In the box, type the message.

Click the ellipsis button to open XPath Builder and create an XPath expression to include in the message. At run time, the evaluated value of the expression is inserted in the message.

**Presentation & Data properties**

Properties for displaying an asset to users. For information about how to present assets and data to users, see Designingdata capture and presentation.

To specify the type of file to display, select one of the following options:

**Use An Application Asset:**

Select to display an application asset, such as a form or Guide, to the user.

**Use A Document Variable:**

Select to display the document that is stored in a document variable.

*Application Asset*

Properties for specifying the asset and data to display to users. These properties appear when Use An Application Asset is selected.

**Asset:**

The presentation asset to display to the user. Click the ellipsis button and select an asset from any application.

**Action Profile:**

The action profile to use with the asset. Select the action profile from the list. The action profiles that appear are already created for the asset that you selected.

**Initial Task Data:**

The data to merge with the asset. Select an xml variable that contains the data. If the action profile that you selected includes a prepare data process, the variable is passed to that process before merging with the asset.

**Prepare Data Process Properties:**

The property group that appears when the prepare data process of the selected action profile includes input or output variables. If the action profile that you selected includes a prepare data process, specify when the process runs:

- The user opens the task: When the user opens the task from their To Do list
- The user opens a draft task: When the user opens a draft task that they previously saved
- The user opens a completed task: When the user opens a task from their task history.

**Render Process Properties:**

The property group that appears when the render process of the selected action profile includes input or output variables. Provide values for the properties that appear.

**Submit Process Properties:**

The property group that appears when the submit process of the selected action profile includes input or output variables. Provide values for the properties that appear.

*Document Variable*

Property for specifying the document variable that holds the file to display to the user. This property group appears when Use A Document Variable is selected.

**Variable:**

Select the document variable to use.

**TIP:** If the variable that holds the file is not yet created, click the plus sign (+) button to create a variable. Later, configure your process so that the variable is populated with a value.

*Reader Submit*

Configure these properties if users are opening tasks using Adobe Reader.

**Submit Via Reader:**

Select if users open and submit tasks using Adobe Reader. When selected, a submit button is displayed in Workspace. If users open tasks using Acrobat Professional and Acrobat Standard, the PDF form includes one or more submit buttons. Workspace detects the buttons and displays them accordingly.

**Submit As:**

If Submit Via Reader is selected, specify the type of data that is submitted when users complete their tasks.

**Output properties**

Properties for storing the data that is sent to the AEM forms Server when users complete tasks. Saving the data is necessary if you want to use the data later in the process.

**TIP:** For the following properties, if the variable required to store data is not yet created, click the plus sign (+) button to create a variable. If you are using an XPath expression, click the ellipsis button to open XPath Builder.

*Task Result*

(Optional) The location in the process data model to store the submitted task data. This value contains information such as the field data from the form or Guide, the submit option that was selected, and task attachments. The data type is `Task Result`. For example, select an existing Task Result variable to store the results in that variable.

*Output Data*

(Optional) The location to store the field data that was submitted with the task. This data is also stored in the Task Result value. It is convenient to store the field data separately to easily pass the data to subsequent Assign Task operations.

The data value is `document` or `xml`. Store the data as a document or an xml value, depending on how you use it later in the process. For example, to use the data as input for another operation, store it in the format that the operation requires.

**NOTE:** If you save task data in a variable that already stores data, the existing data is replaced.

*Task Result Collection*

Properties for storing the submitted Task Result value in a Task Result Collection variable. Storing the task data in a collection is useful for evaluating data that is submitted from a series of different tasks. (See Workingwith captured data.)

**Variable:**

(Optional) The Task Result Collection variable in which to store the Task Result value that is submitted.

**Include Captured Data:**

Select to include field data from the task in the Task Result value that is stored in the collection. If you are interested only in task metadata, do not select this option. For example, do not select it if you are evaluating only the user actions that were clicked for each task.

**Include Attachments:**

Select to include task attachments in the Task Result value that is stored in the collection.

**Workspace User Interface properties**

Properties for specifying which task tools appear in Workspace when users open the task. If you configure these properties using a variable, it must be a Task Runtime UI variable. To provide literal values, specify values for the following properties.

**Default:**

Select to display the standard Workspace features.

**Approval Container (Deprecated):**

Select to display features for reviewing documents. The features enable users to see the status of the review, add comments, and see the task instructions..

**Custom:**

Select to display a custom set of tools. Custom tools are created by using the AEM Forms SDK. (See Programming with AEM Forms.) Click Browse to specify where the executable files are located on the AEM Forms Server.

*UI Options*

Properties for specifying how the form is opened.

**Form Must Be Displayed When Completing:**

Select when users must open the task before completing it. If you do not select this option, users can complete the task from their To Do list without opening the form.

**Open Form Full-screen:**

Select to display the form using all the available space in the web browser window. The form obscures the Workspace UI when the user opens the task. Whether this option is selected or not, users can maximize or minimize the display area of the form as needed.

**Task Access Control List (ACL) properties**

Properties of the access control list (ACL) for tasks. ACLs control which Workspace features are exposed to users. For more information, see Configuring access to task functionality.

The ACL includes a default entry, called *<default ACL>,*that applies to all users. Add a user to the list to override the default access control:

- Click Add to search for the user to add to the list.
- Click the ellipsis button to open XPath Builder and create an XPath expression that identifies a user. The expressions must evaluate to one of the following types of information:
  - Global unique identifier (GUID) of the user account, such as `9A7AD945-CA53-11D1-BBD0-0080C76670C0`
  - Login name of the user, such as `atanaka`
  - Canonical name of the user, such as `atanaka.sampleorganization.com`
  - Eemail address, such as `atanaka@sampleorganization.com`
  - Common name, such as `Akira Tanaka`
  - User value that represents the user.
- To remove a user from the list, select the user and click Delete. You cannot remove the default item.

You can control access to the following features:

**Claim:**

Claim a task. When users have this permission, they can claim a task from another user’s queue.

**Add Notes:**

Add notes to the task. In addition, the user can set read, modify, and delete rights on each new note.

**Share:**

Share the task. When a task is shared, the original permissions are enforced, and the user with whom the task is shared can claim the task as their own.

**Forward:**

Delegate the task to another user.

**Add Attachments:**

Add attachments to the task. In addition, the user can set read, modify, and delete rights on each new attachment.

**Consult:**

Consult the task. Consulting is similar to forwarding the task, but the consultant cannot complete the task. The consultant can only open the task, save the task, add attachments and notes, and return the task to the user who requested the consult.

**Add ACL For Shared Queue:**

Add permissions for other users who have shared queue access to the assigned user's queue.

**Reassignment Restrictions properties**

Properties that control who can be forwarded a task or used as a consultant on a task. For more information, see Configuring task delegations and consultations.

*Forward and Consult*

(Optional) A `TaskDelegate and Consult` value that imposes limitations on who the assigned user can forward their task to or consult with. You specify a user group to indicate who can be used.

If you provide a literal value, specify values for the following properties.

**Forward To And Share With Only Members Of This Group:**

Select this option to limit the forwarding or sharing of the task to a specific user group. Click Browse to select the group. (See About Select Group.)

**Consult Only To Members Of This Group:**

Select this option to limit who can be sent the task for consultation to the members of a user group. Click Browse to select the group. (See About Select Group.)

### Attachments properties

Properties that specify whether attachments and notes are exposed in Workspace. You can also specify where the notes and attachments from the task are saved when the task is submitted. For more information, see Configuring attachments and notes.

**Show Attachment Window For This Task:**

Select to allow users to see task attachments and notes.

**Input List:**

(Optional) A List of document values that represent the attachments and notes to add to the task.

**Output Attachments:**

(Optional) The location to store notes and attachments that are submitted with the task. The data type is a `list` and a subtype of `document`.

**NOTE:** Notes and attachments are saved in the Task Result value that is saved as task output. Use this property to save notes and attachments if the task result is not saved but you need the notes and attachments. (See Output.)

Select a list variable to store the attachments and notes. If you are using an XPath expression, click the ellipsis button to open XPath Builder. If items exist in the list that you specify, new items are appended to the end of the list. To replace the items in a list, specify the index of the first item in the list (for example `/process_data/listVar[1]`).

**TIP:** If the variable for storing notes and attachments data is not yet created, click the plus sign (+) button to create a variable.

### Priority properties

(Optional) A `Task Priority` value that specifies a priority of `Highest`, `High`, `Normal`, `Low`, or `Lowest`. The default is `Normal`. When specifying a literal value, select the priority from the list.

For more information, see Specifying task priority.

### Reminders properties

Properties for configuring task reminders. You can specify the following characteristics of reminders:

- When the first reminder is sent
- The duration between subsequent reminders
- The task instructions when the reminder occurs

For more information, see Sending reminders about tasks.

To specify the value as a variable or XPath expression, a `TaskReminder` value must be provided. To provide a literal value, provide values for the following properties:

**Enable First Reminder:**

Select this option to send a reminder to the task owner.

**Use Business Calendar:**

- Select to use business days instead of calendar days to calculate reminder dates. When this option is selected, the Hours and Minutes boxes are unavailable, and the business calendar associated with the selected user is used. If the selected user does not have a business calendar configured, the default business calendar is used.
- **Days, Hours, and Mintues:**Specify the amount of time (in days, hours, and minutes) after the task is first assigned when the reminder occurs. If the Days, Hours, and Minutes boxes all have a value of `0`, the reminder does not occur.

**Enable Repeat Reminder:**

Select this option to send reminders at regular intervals after the first reminder is sent. The time when repeat reminders occur are calculated from the time that the first reminder is sent.

- **Use Business Calendar:** Select this option to use business days instead of calendar days to calculate repeated reminder dates. When this option is selected, the Hours and Minutes boxes are unavailable, and the business calendar associated with the selected user is used. If the selected user does not have a business calendar configured, the default business calendar is used.
- **Days, Hours, and Mintues:** Specify the amount of time (in days, hours, and minutes) after the task is first assigned when the reminder occurs. If the Days, Hours, and Minutes boxes all have a value of `0`, the reminder does not occur.

**Change Task Instructions On Reminder:**

Select this option to change the task instructions when reminders occur. Type the template for the instructions in the box. Click the ellipsis button to incorporate process data using XPath expressions. For information about templates, see Specifying template expressions.

**Escalation properties**

Properties for reassigning the task to another user after a specific amount of time. Specify when the task is reassigned and who is assigned the task when escalation occurs. For more information, see Escalating tasks.

*Escalate Task*

(Optional) Select this option to escalate the task. If you are using a variable or XPath expression, the value must be a `boolean` value. A value of `true` escalates the task. The default value of `false` does not escalate the task.

*Schedule Escalation*

(Optional) The amount of time that passes after the task is assigned to the initial user until the task is escalated. To provide literal values, configure the following properties:

**Use Business Calendar:**

Select to use business days instead of calendar days to calculate escalation dates. When this option is selected, the Hours and Minutes boxes are unavailable, and the business calendar associated

with the selected user is used. If the selected user does not have a business calendar configured, the default business calendar is used.

**Day, Hours, and Minutes:**

Specify the number of days, hours, and minutes that pass until the escalation occurs. If the Days, Hours, and Minutes boxes all have a value of `0`, the task is not reassigned to another user.

To specify a value by using a variable or XPath expression, the value must be a `TaskDate` value.

*Select Escalation User*

(Optional) Select one of the following options to specify who to reassign the task to when the time frame specified in Schedule Escalation property is reached.

**Allow Out Of Office Designation:**

Select this option to reassign the task according to the escalation user's out-of-office preferences.

**Assign To Specific User:**

Select this option to specify a user to assign the task to when the escalation occurs. The user must exist in the AEM Forms environment. Use the Browse button to open the Select User dialog box and select the user profile. (See About Select User.)

**Assign To Process Creator:**

Select this option to specify that the user who started the process is assigned the task when escalation occurs.

**Assign To Group:**

Select this option to assign the task to a user group when escalation occurs. Users that belong to the group can be assigned a task in one of the following ways:

- **Assign To Group Queue:**Assigns the task to the group's shared To Do list.
- **Assign to Random User In Group:** Assigns the task to any user who belongs to the specified group profile.

  Use the Browse button to open the Select Group dialog box and select the group to assign the task. (See About Select Group.)

**XPath Expression:**

Select this option to create an XPath expression that evaluates to one of the following values:

- A `string` value that represents a user or group GUID, canonical name, login identification, email address, or common name. Use a common name or email address only if you are certain that they are unique.
- A `User` value.
- A `Group` value.

**TIP:** You can obtain User and Group values by using the User Lookup service. Use the User Lookup service in the process before the Assign Task operation. Save the results in a variable so that it is available in the XPath expression. (See UserLookup.*)*

### Deadline properties

Properties for configuring task deadlines. You can specify when the deadline occurs, whether to change task instructions when the deadline occurs, and whether a specific route is followed. For more information, see Setting deadlines for tasks.

To configure deadlines by using a variable or XPath expression, a `Task Deadline` value must be provided. To provide a literal value, specify values for the following properties:

**Enable First Deadline:**

Select this option to create a deadline for the task.

- **Use Business Calendar:** Select this option to use business days instead of calendar days to calculate deadline dates. When this option is selected, the Hours and Minutes boxes are unavailable, and the business calendar associated with the assigned user is used. If the user does not have a business calendar configured, the default business calendar is used.
- **Days, Hours, and Mintues:** Specify the amount of time (in days, hours, and minutes) after the task is first assigned when the deadline occurs. If the Days, Hours, and Minutes boxes all have a value of `0`, the deadline does not occur.

**Change Task Instructions On Reminder:**

Select this option to change the task instructions when reminders occur. Type the template for the instructions in the box. Click the ellipsis button to incorporate process data by using XPath expressions.

**Follow A Specific Route On Deadline:**

Select this option to determine the next operation to execute by specifying the route to follow when the deadline occurs.

**Select Route:**

Select the route to follow when Follow A Specific Route is selected. If no routes originate from the Assign Task operation, There Are No Outbound Routes appears in the list.

### Custom Email Templates properties

Properties for configuring the text in emails that are sent when a task is assigned, a reminder occurs, or a deadline occurs. In the list, select the event and then configure the email template properties:

**Use Server Default:**

Use the email template configured on the server.

**Do Not Send Email:**

Do not send an email notification.

**Customize:**

Modify the email template for this task. Click Edit Email Template to use the Email Template Editor dialog box to modify the email template for the specified action. (See AboutEmail Template Editor.)

**Exceptions**

The Assign Task operation can throw an InvalidPrincipal exception. The exception occurs when a user or group that no longer exists is specified as a property value.

**Assigning tasks across development and production environments**

Task assignments to a user or user group fail even though the user domains in the development and production environments are configured exactly the same.

If a process that was tested in a development environment is executed in a production environment, the task assignment to a user or group fails even though the user domains in the development and production environments are configured exactly the same. The process is also not transferable to another computer (production environment) that had the same user or group added in the administration console. To resolve this issue, do one of the steps below:

- **Use the Assign Multiple Tasks operation**: Create a UserList and add the group you have created to the UserList. In the User 2.0 service, select the UserList you created as the source of participants for that Assign Multiple Tasks operation.
- **Use the XPath option in the User 2.0 service**: Provide either the name or email address of the group you have created. The Xpath expression resolves to the following (including the quotes):
  - “Group Name”
  - “groupEmail@server.com”
- **Production environment**: Deploy the process, and then use Workbench to connect to the AEM Forms Server. Check out and open the process, and then reconfigure the Initial User Selection properties. For example, if you have Assign To Specific User selected, browse and select a different user, and then browse and select the original user again. This sets the Initial User Selection properties to the correct user in the production environment.

RELATED LINKS:

Configuring the presentation of task data

Assets for capturing and presenting data

About business calendars

## User exceptions

The User service provides the following exception for throwing exception events.

**InvalidPrincipal**

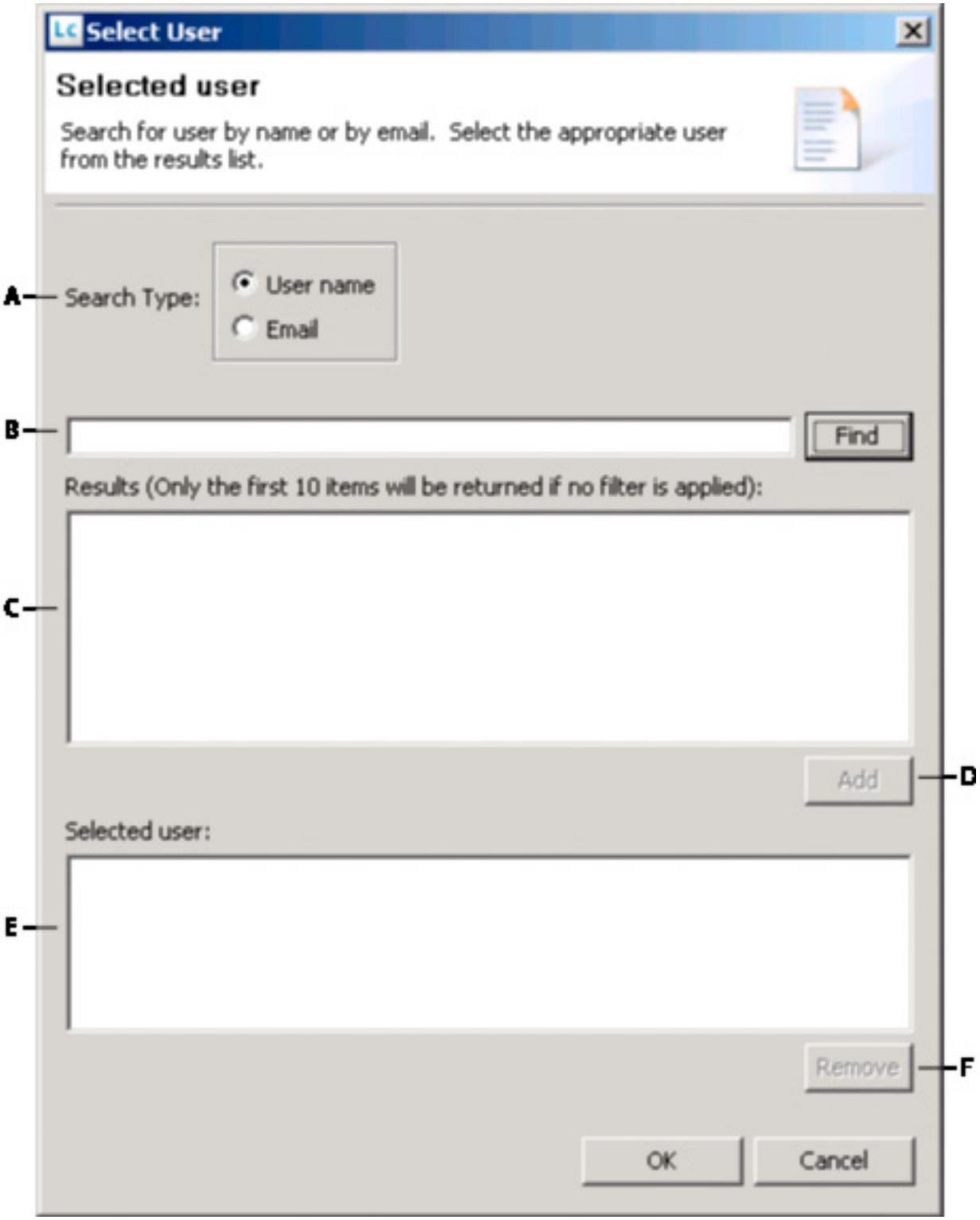
Thrown when the task cannot locate a specified user or group profile. This exception can occur for the following reasons:

- A user is deleted or disabled locally or in LDAP.

- The domain that a user is part of is deleted or disabled locally or in LDAP.
- The group to which your user belongs is deleted or disabled locally or in LDAP.

## About Select User

Use the Select User dialog box to choose the user to assign the task to.




**A.**

Search by user name or email.

**B.**

Search results.

**C.**

Filter search results by the specified string.

**D.**

Add selected user or group.

**E.**

List of principals that will be added.

**F.**

Remove principal from Selected User pane.

## About Select Group

Use the Select Group dialog box to choose the group to assign the task to.

**Search Type**

Search by user name or email.

**Search Filter**

Filter search results by the specified string.

**Results**

Displays the search results.

**Add**

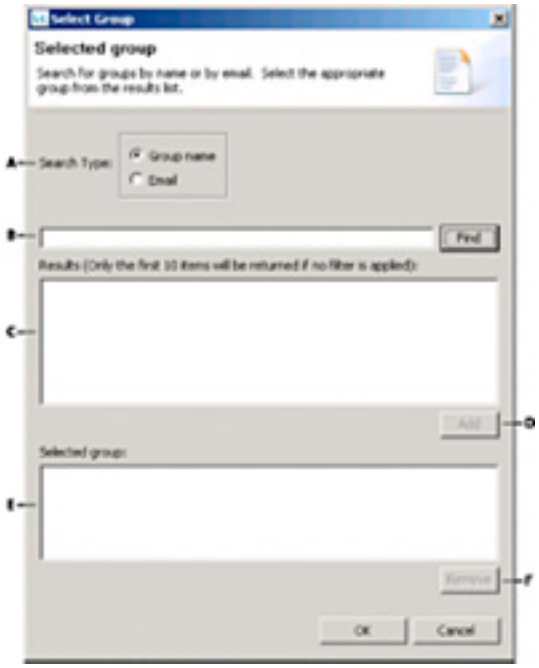
Add selected user or group.

**Selected user**

List of principals that will be added.

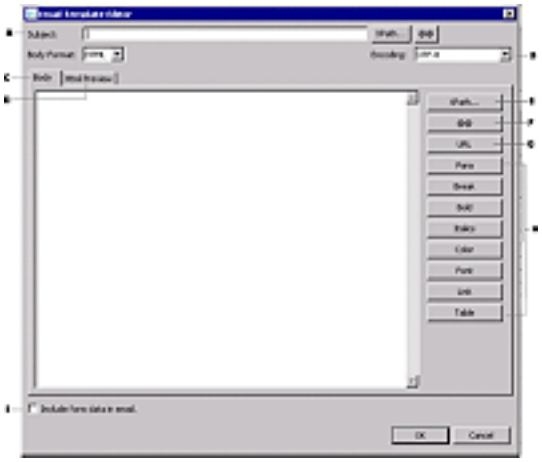
**Remove**

Remove the user selected in the list.




## About Email Template Editor

Use the Email Template Editor to override the default email template for the specific task.



**A.**

The subject for the email. You can include text, XPath expressions, or forms workflow variables.

**B.**

Buttons for HTML tags available for HTML-formatted email templates.

**C.**

Encoding for the email template.

**D.**

Editing area for contents of email template.

**E.**

Preview area for HTML-formatted email templates.

**F.**

Add an XPath expression.

**G.**

Add an URL to the email template that opens the task.

**H.**

Include an attachment of the form and its data with the email message.

**I.**

Add a forms workflow variable from the Variable Picker dialog box.

**Variable Picker:**

The forms workflow variable to add.

- **@@taskid@@**: The unique identifier for the current task.
- **@@instructions@@**: The text that is entered for the task.
- **@@notification-host@@**: The host name of the AEM Forms Server.
- **@@description@@**: The contents from the Description property for the Assign Task operation.
- **@@process-name@@**: The name of the process.
- **@@operation-name@@**: The contents of the Name property from the Assign Task operation.

**Body Format:**

The format the email is sent in. Choose between HTML or TEXT. The default format is HTML.

**:**

The default encoding format used for the email. The default is UTF-8. It is recommended that ISO-2022JP be used for Japanese character sets.

Select an encoding format from the list:

**NOTE:** The list of defaults depends on the operating system you use.

**Include Form Data In Email:**

The default is not to include form data as an attachment to the email.

# 22.51. User Lookup

Enables processes to search for user accounts and groups that are stored in User Manager. For example, you can find a specific user at run time and then configure an Assign Task operation to assign a task to that user.

The operations that the User Lookup service provides return User and Group data types. After retrieving the user or group, you can get attribute values of user or group accounts. For example, you can get the email address of a user to use with the Email service.

**NOTE:** When searching for multiple users or groups, the maximum number of users or groups that are returned is 1000.

For more information about User Manager, see User Manager Help

.

## Find Group operation

Retrieves information about a user group from User Manager based on search filters that you provide. The search returns the first group that is found that meets the search criteria. The search also returns the total number of groups that meet the criteria.

Use the FindGroups operation to retrieve multiple groups that meet the same search criteria.

For information about the General and Route Evaluation property groups, see Commonoperation properties.

### Filter properties

Properties for specifying search criteria. You must specify at least one filter. You can specify more than one filter to narrow the search results.

*Domain Name*

A `string` value that represents the User Management domain that the groups belong to. To specify a literal value, select a domain from the list.

For information about domains, see Setting up and managing domains in AEM Forms administration help

.

*Universal ID*

A `string` value that represents the unique identification of the group. This value searches the Unique Identifier property that is configured in the Directory Groups properties of the domains that are created in User Management.

You must provide the entire identification, for example `fb5f4c02-1dd111b2-8093fe29-7dd5631b`.

*Name*

A `string` value that represents the name of the group. This value searches the Full Name property that is configured in the Directory Groups properties of the domains that are created in User Management.

You can specify all or part of the name for the group that you want to retrieve.

**Result properties**

Properties for specifying where to store operation results.

*Results*

An `int` value that stores the number of groups that were found that matched the search filters. The maximum value that Result can be is `10`. After ten matches are found, the search is stopped to conserve server resources.

*Group*

A `Group` value that represents the first group that was found that matches the search filters.

**Exceptions**

The exception event attached to this operation can receive UMException exceptions.

## Find Groups operation

Retrieves information about one or more groups from User Manager based on search filters that you provide. The search returns all users that are found that meet the search criteria.

**NOTE:** The maximum number of groups that is returned is 1000.

To retrieve only one group that meets the search criteria, use the FindGroup operation.

For information about the General and Route Evaluation property groups, see Commonoperation properties.

**Filter properties**

Properties for specifying search criteria. You must specify at least one filter. You can specify more than one filter to narrow the search results.

*Domain Name*

A `string` value that represents the User Management domain that the groups belong to. To specify a literal value, select a domain from the list.

For information about domains, see Setting up and managing domains in AEM Forms administration help

.

*Universal ID*

A `string` value that represents the unique identification of the group. This value searches the Unique Identifier property that is configured in the Directory Groups properties of the domains that are created in User Management.

You must provide the entire identification, for example `56e04201-1dd211b2-80cafe29-7dd5631b`.

*Name*

A `string` value that represents the name of the groups. This value searches the Full Name property that is configured in the Directory Groups properties of the domains that are created in User Management.

You can specify all or part of the name for the groups that you want to retrieve.

**Result properties**

Properties for specifying where to store operation results.

*List of Groups*

The location to store the returned `list` value that contains `Group` values. Each `Group` value represents a group that matches the search filters. For example, to save the list in a variable, select a `list` variable from the list.

For information about retrieving values from a list, see Accessingdata in data collections.

*Output XML Document*

The location in the process data model to store a returned document value that represents an XML document that contains the search results. The XML represents a `list` of `Group` values. For example, the following XML is the result of a search that returns the default All Principals group:

```
<?xml version="1.0" encoding="UTF-8"?>
<groups>
<group>
<principalOid>B98E6F7F-5DC7-102C-A430-00000A24CA8A</principalOid>
<domainName>DefaultDom</domainName>
<commonName>All Principals</commonName>
<canonicalName>GROUP_ALLPRINCIPALS</canonicalName>
</group>
</groups>
```

**Exceptions**

The exception event attached to this operation can receive UMException exceptions.

## Find User operation

Retrieves information about a user account from User Manager based on search filters that you provide. The search returns the first user that is found that meets the search criteria. The search also returns the total number of users that meet the criteria.

Use the FindUsers operation to retrieve multiple users that meet the same search criteria.

For information about the General and Route Evaluation property groups, see Commonoperation properties.

### Filter properties

Properties for specifying search criteria. You must specify at least one filter. You can specify more than one filter to narrow the search results.

*Exact Match*

A boolean value that determines whether users are returned only when their properties exactly match the Name and Email filters. A value of true indicates an exact match is returned. A value of false indicates the first user found that partially matches the search criteria is returned.

If you are expressing a literal value, select Exact Match to match the entire user name or email address.

*Domain Name*

A `string` value that represents the User Management domain that the user belongs to. To specify a literal value, select a domain from the list.

For information about domains, see Setting up and managing domains in AEM Forms administration help

.

*Universal ID*

A `string` value that represents the unique identification of the user. This value searches the Unique Identifier property that is configured in the Directory Users properties of the domains that are created in User Management.

You must provide the entire identification, for example `56e04201-1dd211b2-80cafe29-7dd5631b`.

*Name*

A `string` value that represents the name of the user. This value searches the Full Name property that is configured in the Directory Users properties of the domains that are created in User Management.

You can specify all or part of the name for the user that you want to retrieve.

*Email*

A `string` value that represents the email address of the user. This value searches the Primary Email property that is configured in the Directory Users properties of the domains that are created in User Management.

You can specify all or part of the email address for the user that you want to retrieve.

*Login Id*

A `string` value that represents the login name that the user logs in with. This value searches the Login ID property that is configured in the Directory Users properties of the domains that are created in User Management.

You can specify all or part of the login name for the user that you want to retrieve.

**Result properties**

Properties for specifying where to store operation results.

*Results*

The location to store the returned `int` value that stores the number of users that were found that matched the search filters. The maximum value that Result can be is 10. After ten matches are found, the search is stopped to conserve server resources.

*User*

A `User` value that represents the first user account that was found that matches the search filters.

**Exceptions**

The exception event attached to this operation can receive UMException exceptions.

## Find Users operation

Retrieves information about one or more user accounts from User Manager based on search filters that you provide. The search returns all users that are found that meet the search criteria.

**NOTE:** The maximum number of users that is returned is 1000.

To retrieve only one user that meets the search criteria, use the FindUser operation.

For information about the General and Route Evaluation property groups, see Commonoperation properties.

**Filter properties**

Properties for specifying search criteria. You must specify at least one filter. You can specify more than one filter to narrow the search results.

*Exact Match*

A boolean value that determines whether users are returned only when their properties exactly match the Name and Email filters. A value of true indicates that only exact matches are returned. A value of false indicates that user accounts that partially match the search criteria are returned.

If you are expressing a literal value, select Exact Match to match the entire user name or email address.

*Domain Name*

A `string` value that represents the User Management domain that the users belong to. To specify a literal value, select a domain from the list.

For information about domains, see Setting up and managing domains in AEM Forms administration help

.

*Universal ID*

A `string` value that represents the unique identification of the user. This value searches the Unique Identifier property that is configured in the Directory Users properties of the domains that are created in User Management.

You must provide the entire identification, for example `56e04201-1dd211b2-80cafe29-7dd5631b`.

*Name*

A `string` value that represents the name of the users. This value searches the Full Name property that is configured in the Directory Users properties of the domains that are created in User Management.

You can specify all or part of the name for the users that you want to retrieve.

*Email*

A `string` value that represents the email address of the users. This value searches the Primary Email property that is configured in the Directory Users properties of the domains that are created in User Management.

You can specify all or part of the email address for the users that you want to retrieve.

*Login ID*

A `string` value that represents the name that the user logs in with. This value searches the Login ID property that is configured in the Directory Users properties of the domains that are created in User Management.

You can specify all or part of the login names for the users that you want to retrieve.

**Result properties**

Properties for specifying where to store operation results.

*List of Users*

The location to store the returned `list` value that contains `User` values. Each `User` value represents a user account that matches the search filters.

For information about retrieving values from a list, see Accessingdata in data collections.

*Output XML Document*

The location in the process data model to store a returned document value that represents an XML document that contains the search results. The XML represents a `list` of `User` values. For example, the following XML is the result of a search that returns the default Administrator user:

```
<?xml version="1.0" encoding="UTF-8"?>
<users>
<user>
<principalOid>BA79A041-5DC7-102C-8ED7-00000A24CA8A</principalOid>
<userID>administrator</userID>
<domainName>DefaultDom</domainName>
<familyName>Administrator</familyName>
<givenName>Super</givenName>
<initials/>
<commonName>Super Administrator</commonName>
<canonicalName>SuperAdmin</canonicalName>
<telephoneNumber/>
<postalAddress/>
<email/>
<org/>
</user>
</users>
```

**Exceptions**

The exception event attached to this operation can receive UMException exceptions.

## findGroupMembers operation

Retrieves the members of a group that is defined in User Manager. Members are users and groups that belong to the group.

You can retrieve all users of the group, or a subset.

Use the Offset and resultSize properties to retrieve a subset of the user members. Groups include a list of users in a specific order. The Offset property specifies the index of the first user to include in the subset. The resultSize property specifies the number of users to include. Indexes of members are zero-based.

**NOTE:** The Offset and resultSize properties affect only the users that are returned. All groups that are members of the targeted group are always returned.

For example, consider a group with seven members `user0` through `user6`. A value of 2 is used for the Offset property, and a value of 3 for the resultSize property. The operation returns the third, fourth, and fifth members of the group, `user2` through `user4`.

For information about the General and Route Evaluation property groups, see Commonoperation properties. In addition, this operation provides the following property groups and exceptions:

Input

Output

Exceptions

**Input properties**

Properties for specifying the group and members to retrieve from the group.

*Group*

A `Group` value that represents the User Manager group to retrieve members from.

*Offset*

An `int` value that specifies the index of the first user to include in the subset of users to return. The index of the member list is zero-based.

*resultSize*

An `int` value that specifies the number of users to retrieve. Its default values are 10 and 1000 respectively.

**Output properties**

Properties for specifying where to store operation results.

*Groups*

The location to store the returned `list` of `Group` values. Each group value represents a group that is retrieved from the searched group. For example, to save the results in a variable, select a list variable that stores Group values.

*Users*

The location to store the returned `list` of `User` values. Each `User` value represents a user account that is retrieved from the group. For example, to save the results in a variable, select a list variable that stores User values.

For information about retrieving values from a list, see Accessingdata in data collections.

**Exceptions**

The exception event attached to this operation can receive UMException exceptions.

## User Lookup Exceptions

### UMException

Thrown when an error occurs when interacting with User Manager. This exception can be thrown when User Lookup operations are not configured correctly, for example when no search filters are configured.

# 22.52. Variable Logger

Enables processes to send messages about variable values to the system log or to log files on the file system of the AEM Forms Server.

For information about using the Variable Logger service, see Services Reference for AEM Forms

## log operation

Logs messages about process variables to system resources or saves them to a text file:

- Messages logged to system resources are reported using either the application server log or Standard Out. The configuration of the application server determines where the server log is saved, and what system resource is used for Standard Out. You can set the type of information that is logged when you use system logging.
- Messages logged to a text file are saved in a file on the file system of the AEM Forms Server.

  *When the Log operation is executed, the current values of all process variables are logged. For example, the following items were added to the system log for a process that includes the `string` variable named `stringVar`:*

  ```
  2008-04-10 16:54:30,718 WARN
  [com.adobe.idp.dsc.variablelogger.VariableLoggerService] [PID:7]
  /process_data/@stringvar - String:  This is the value of the string
  2008-04-10 16:54:30,718 WARN
  [com.adobe.idp.dsc.variablelogger.VariableLoggerService] Variable
  Properties:
  VP1207841667573:length = [100]
  ```

For information about the General and Route Evaluation property groups, see Commonoperation properties.

### System Logging properties

Properties for logging variable values to system resources.

*Log To System Logger*

Select to have messages about variables sent to the application server log.

*Log To Standard Out*

Select to have messages about variables sent to Standard Out.

*Logging Level*

The severity level that you want to use for the log items in the system log. You can select one of the following values.

- Off (no information is logged)
- Debug
- Info
- Warning
- Severe

  *For example, when Info is selected, log entries for a the `stringVar` variable value are logged as INFO messages in the JBoss log:*

  ```
  2008-04-10 16:53:53,937 INFO
  [com.adobe.idp.dsc.variablelogger.VariableLoggerService] [PID:6]
  /process_data/@str4ingvar - String:  The string value.
  ```

  *When Warning is selected, the log entries in the JBoss log are logged as WARN messages:*

  ```
  2008-04-10 16:54:30,718 WARN
  [com.adobe.idp.dsc.variablelogger.VariableLoggerService] [PID:7]
  /process_data/@str4ingvar - String:  The string value
  ```

**Directory Logging properties**

Properties for logging information to text files on the file system.

*Log To A directory*

Select to have messages about variables saved to files on the file system.

*Directory Path*

A `string` value that represents the directory in which to save the log files. If the directory does not exist, it is created automatically.

*File Name Prefix*

A `string` value that represents the prefix to use for the file name in which the log is saved. The file name extension.log is appended to the prefix.

*Logging Mode*

Specifies whether information for each new process instance overwrites previous log entries. You can select one of the following values.

**Overwrite:**

The information in the log file is overwritten when messages for a new instance are logged.

**Append:**

Information for each process instance is logged to the same file, appending new information to the end of the file.

**CreateNew:**

A new log file is created for each process instance. A unique prefix is added to each log file name.

### Exceptions

The exception event attached to this operation can receive Java.io.IOException, VariableLoggerConfigurationException, and VariableLoggerExecutionException exceptions.

## Variable Logger exceptions

The Variable Logger service provides the following exceptions for throwing exception events.

### Java.io.IOException

Thrown when a file input/output error occurs on the AEM forms Server.

### VariableLoggerConfigurationException

Thrown when a configuration error occurs on the AEM Forms Server.

### VariableLoggerExecutionException

Thrown when an execution error occurs on the AEM Forms Server.

# 22.53. Wait Point

Enables you to delay the progression of a process at a step in the process.

For information about using the Wait Point service, see Services Reference AEM Forms

.

## businessCalendarWait operation

The businessCalendarWait operation completes a specified amount of time after it is executed using business days instead of calendar days. Use this operation when you want to delay the execution of the next operation for a specified number of business days.

For information about the General and Route Evaluation property groups, see Commonoperation properties.

### Input properties

Properties that specify the number of business days to wait and the business calendar to use.

*Date*

A `Business Calendar Date` value that is used to calculate the date and time to resume execution. If a business calendar is not specified, the default business calendar is used.

If you provide a literal value, you can specify the following parameters.

**Number of Business Days:**

An `int` value that represents the number of business days to wait before resuming execution.

**Business Calendar Name:**

A `string` value that specifies that name of the business calendar to use. The business calendar is defined on the AEM Forms Server. The default is `<Use Default Calendar>`.

### scheduleWait operation

The scheduleWait operation completes a specified amount of time after it is executed. Use this operation when you want to delay the execution of the next operation in the process.

For information about the General and Route Evaluation property groups, see Commonoperation properties.

**Input properties**

Specify the amount of time to wait by providing values for days, hours, minutes, and seconds. The total time is the cumulation of each value.

*Days*

An `int` value that represents the number of days to wait.

*Hours*

An `int` value that represents the number of hours to wait.

*Minutes*

An `int` value that represents the number of minutes to wait.

## 22.54. Web Service

Enables processes to invoke web service operations. This service provides the InvokeWeb Service operation.

For more information about the Web Service service, see Services Reference for AEM Forms.

## Web Service service configuration

Configure the following properties to enable the Web Service service to call web services using the HTTPS protocol. (See Editingservice configurations.)

## Invoke Web Service operation

A web service client that invokes a web service operation and saves the response as process data, including attachments. Invoke Web Service interacts with web services by sending and receiving SOAP messages. For authentication, this operation supports HTTP Basic Authentication and HTTPS authentication.

This operation supports sending inline MIME, SwaRef, and base64, and MTOM attachments with SOAP messages using the WS-Attachment protocol. DIME attachments and MTOM attachments sent as base64-encoded byte array that are embedded in the SOAP XML are not supported.

Use the Invoke Web Service operation to call a web service to retrieve data that your process requires. For example, a purchase order process uses a web service to retrieve the name of a customer, their mailing address, and other billing information.

For information about the General and Route Evaluation property groups, see Commonoperation properties.

### Web Service Options properties

Properties for specifying the web service operation to invoke.

*Options*

A value that represents the options to send to the web service for invoking a service operation.

Use the Web Service Settings dialog box to specify the value. (See AboutWeb Service Settings.) Click the ellipsis button to display the dialog box.

### Web Service Response properties

Properties for saving the web service response message.

*Response*

The location to save the response message that the web service returns as a result of the invocation request. The data type is `xml`. Click the ellipsis button to open XPath Builder to create the XPath expression that resolves to the location.

*Attachments*

The location in the process data model to save documents that are attached to the response message. The data type is `list`, which holds `document` values.

*CDATA List*

The location to save text that is located inside CDATA sections in the web service response. The data type is list which holds document values. The list contains a `document` value for each CDATA section in the web service response.

When you save CDATA selections in a list of documents, you do not have to parse the web service response to access the text. For example, when AEM Forms web services return XML code in their response, it is located in a CDATA section. Other web services can include other types of data inside CDATA sections.

### Exceptions

This operation can throw these exceptions: WebServiceConfigurationException, WebServiceInvocationException, and WebServiceResponseParsingException.

### Web Service exceptions

The Web Service service provides the following exceptions for throwing exception events.

*WebServiceConfigurationException*

Thrown when there is a problem with the configuration of the WebService Options.

*WebServiceInvocationException*

Thrown when an error occurs when the web service is called.

*WebServiceResponseParsingException*

Thrown when an error occurs when parsing the response message that the web service returns.

## About Web Service Settings

The Web Service Settings dialog box lets you create the SOAP message to send to the web service to invoke a web service operation.

After you provide the URL to the web service definition, Web Service Settings interprets the definition and populates other values on the tab. You can then select the web service operation to invoke. Based on the operation that you select, a template of the SOAP request message is generated. You then insert values into the message as required.

Web Service Settings also let you test your invocation request by sending a test message and displaying the response message that the web service sends.

### Settings tab

Use the settings tab to provide a link to the web service definition and to provide the information required to connect to a web service. You can also access web services from a AEM Forms Server. (See Invokingservices.)

*WSDL URL*

The URL of the web service definition. The definition is written in web service definition language (WSDL). WSDLs contain all the information that web service clients require to call web service operations and process the response.

**NOTE:** The AEM Forms Server must be able to access the WSDL URL at run time and at design time.

If you have access to the service, click Load after you enter the value for WSDL URL. Clicking Load causes Web Service Settings to read and interpret the definition. It then automatically populates many of the properties on the Settings tab, such as the Target URL property.

**TIP:** If your development environment has network limitations that prevent you from accessing the web service, save the WSDL file to the file system, and load that file to populate the other property values. After loading the file, you replace the value of the WSDL URL property with the actual URL for the service definition.

*User Name*

The user name of the account that you can use to access the web service. Provide a user name only if the web service requires authentication.

If the user name is saved as process data, you can type the XPath expression that resolves to the location where the user name is stored. Click XPath to open XPath Builder.

*Password*

The password that corresponds with the user name that you provided for the User Name property.

If the password is saved as process data, you can type the XPath expression that resolves to the location where the password is stored. If the expression resolves to a variable, set a default value for the variable to use for testing. Click XPath to open XPath Builder.

*Port*

The service that you want to use, that is exposed through the WSDL URL. A WSDL URL can have multiple services available. After the WSDL URL is loaded, this list is populated with the available ports. (See WSDLURL.)

*Target URL*

The URL that provides access to the web service. After the value for Port is specified, the default target URL as defined in the WSDL is displayed. You can override this target URL value by typing the actual target URL for the selected port.

If the URL is saved as process data, you can type the XPath expression that resolves to the location where the value is stored. If the expression resolves to a variable, set a default value for the variable to use for testing. Click XPath to open XPath Builder.

When you access web services from a AEM Forms Server, append ?wsdl&async=true&lc&lc_version=9.0.0 to the URL when the service is an asynchronous service. Services for long-lived processes are asynchronous.

*Operation*

The web service operation to invoke. The list is populated automatically if you loaded the web service definition. (See WSDLURL.)

*Time Out (in secs)*

The amount of time that you want to wait for a response from the web service before abandoning the web service invocation. The value you provide is in seconds.

If the time-out value is saved as process data, you can type the XPath expression that resolves to the location where the value is stored. If the expression resolves to a variable, set a default value for the variable to use for testing. Click XPath to open XPath Builder.

*WSDL Content*

The WSDL that is retrieved from the URL provided for the WSDL URL property. The WSDL appears when you click the Load button.

*Embed WSDL*

Select this option to embed the WSDL in the process. When selected, the embedded WSDL is used when the invoke operation executes at run time. The embedded WSDL is updated each time you click the Load button.

If this option is not selected, the WSDL is retrieved from the WSDL URL at run time.

**HTTP Settings tab**

Use the HTTP Settings tab to configure settings related to the HTTP protocol.

**Send Authentication Headers Without First Receiving An Authentication Challenge:**

Select when the web service does not send a challenge request for authentication.

**Request tab**

Use the Request tab to create the SOAP request message that is sent to the web service to invoke the operation.

*SOAP Request*

The SOAP message to send to the web service at run time. For more information about SOAP messages, see AboutSOAP invocation messages.

If you loaded the WSDL and selected the port and operation to invoke, click Generate to populate the editing box with a template message. (See WSDLURL.) The template message is complete except for the values that you must provide for the required operation parameters. Question marks (?) indicate values that you must provide.

**TIP:** Click Remove '?' to remove all the question marks from the SOAP request.

If the operation has optional parameters that you want to provide values for, select Include Optional. This option includes the XML for optional parameters in the message.

If you cannot load the service definition, enter the message manually.

To use values in process data for operation parameters, use XPath expressions that evaluate to the data location. XPath expressions must appear inside braces and between dollar signs, as in `{$expression$}`. Click XPath to open XPath Builder. The expression that you create with XPath builder is inserted at the location of the cursor in the editing box.

The following example template message shows a request to invoke a web service's `invoke` operation. The values for the parameters `intvar` and `strvar` have a question mark as placeholder values.

```
<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:ser="http://adobe.com/idp/services">
<soapenv:Header/>
<soapenv:Body>
<ser:invoke>
<ser:intvar>?</ser:intvar>
<ser:strvar>?</ser:strvar>
</ser:invoke>
</soapenv:Body>
</soapenv:Envelope>
```

*SOAP Request For Test*

The SOAP request to send to the web service for testing at design time.

As with the SOAP Request property, if you have loaded the web service definition, click Generate to populate the editing box with a template message. (See WSDLURL.) The template message is complete except for the values that you must provide for operation parameters. Question marks (?) indicate values that you must provide.

**TIP:** Click Remove '?' to remove all the question marks from the SOAP request.

If the message includes XPath expressions as parameter values, the expressions must resolve to process variables that are configured with a default value. Otherwise, replace the parameters with literal values for testing.

You use the Test tab to test the message. (See Test.)

### Attachment tab

Use the Attachments tab to add file attachments to the SOAP message. You can add attachments only if the web service definition indicates that they are allowed or required. Also, the file to attach must already be saved as process data so that you can reference it.

**NOTE:** To add required attachments, you must have already loaded the WSDL, selected the port and operation, and defined the SOAP request message. (See WSDL URL.)

Each row in the table represents a file attachment:

- Click Load Attachment Part to add rows to the table for the attachments that must be specified for the web service operation that you are invoking. The Part and Type columns are populated with values automatically.
- Click Add Attachment Part to add rows to the table for optional attachments that the WSDL does require. The default value for Part is <anonymous>. You can change the value as required. When you click the button, XPath Builder opens so that you can create an expression that evaluates to the document to attach.
- Select a row and click Remove Attachment Part to remove the attachment.

For each attachment, you must provide values for Attachment and Content-type:

**Attachment:**

Click the cell, and then click the ellipsis button that appears to open XPath Builder. The XPath expression that you create must resolve to the location where the attachment is saved.

**Content-Type:**

Type the MIME type of the file that you are attaching, for example `application/pdf`.

**Test tab**

Use this tab to test the SOAP message that you created for testing on the Request tab. To test the message, the message is sent to invoke the web service operation that you previously specified.

Click Test to send the test message to invoke the web service operation. The response message that the web service returns appears in the text area.

**NOTE:** To test, you cannot use XPath Expression as property values because the process data that they resolve to does not yet exist at design time.

RELATED LINKS:

UsingTable data types for SAP web services

Invokingservices

InvokeWeb Service

WebService Options

## About SOAP invocation messages

SOAP is a protocol for exchanging XML-based messages over a network, and is used for sending messages between web services and their clients. Client software sends SOAP messages to web services to invoke web service operations. Web services return the operation results to clients using another SOAP message.

The XML that represents a SOAP message that is sent to invoke a web service operation is comprised of the message envelope, header, and body:

**Envelope:**

The Envelope element is the root element of the XML document that represents the SOAP message. This element contains the message header and body.

**Header:**

The Header element contains information that is not directly related to the web service operation. This element serves as a mechanism for providing information to the software that processes the message. For example, the header is typically used to transfer user credentials for authentication reasons. It is optional to include the Header element unless the web service requires it.

**Body:**

The Body element contains one element that represents a call to the web service operation that is being invoked. The element type is the name of the operation. The operation element contains elements that represent the operation parameters. These element types are the names of the parameters, and contain the parameter values.

The name of the web service operations, as well as the parameter names and data types are defined in the web service definition. Most web service clients retrieve and interpret the definition automatically.

**Namespaces**

The Envelope element must be prefixed with a namespace alias whose declaration uses `”http://schemas.xmlsoap.org/soap/envelope/”` as the namespace identifier. If the following namespace declaration is used for this namespace, the Header element is prefixed with `soapenv`:

```
xmlns:soapenv=”http://schemas.xmlsoap.org/soap/envelope/”
```

Other namespaces that are declared are namespaces that the web service definition defines. These namespaces are used to prefix the XML elements that are associated with the web service operation. For example, web services for invoking activated processes use the namespace identifier `”http://adobe.com/idp/services”`.

**Example**

The following XML code is an example SOAP message element that represents a call to the web service’s operation named `operation_name`. The operation takes one parameter named `parameter_name`.

```
<soapenv:Envelope
xmlns:soapenv=”http://schemas.xmlsoap.org/soap/envelope/”
xmlns:ser=”http://foo.net/services”>
<soapenv:Header />
<soapenv:Body>
<ser:operation_name>
<ser:parameter_name>?</ser:parameter_name>
</ser:operation_name>
</soapenv:Body>
</soapenv:Envelope>
```

## Invoking services in AEM Forms using Web Services

You can invoke services that are part of AEM Forms using web services. For instance, you can invoke a service on another AEM forms Server using the Invoke Web Service operation in your process. Invoke the service on another AEM Forms Server when the service is not available on the AEM Forms Server that executes the process. When you choose to invoke web services, the default for attachments is MTOM.

**TIP:** For a service in AEM Forms, you determine the WSDL URL to use by accessing the web service endpoint information in Applications and Services. (See *Applications and Services*administration help

*.) For example, to determine the WSDL URL to use for the Generate PDF service, in administration console, select Applications and Services > GeneratePDFService. In the Endpoints tab, click GeneratePDFService for the provided Default SOAP Endpoint to see the WSDL URL.*

To invoke asynchronous service operations from AEM Forms, add the suffix `?wsdl&async=true&lc&lc_version=9.0.0` to the value of Target URL after the WSDL is successfully loaded. If you do not add `?wsdl&async=true&lc&lc_version=9.0.0` to the value, an Operation Not Found exception occurs when you invoke the web service.

**TIP:** Services for long-lived processes are asynchronous.

For example, if you want to use an asynchronous operation from the Generate PDF service, perform the following steps. In the following example,*[servername]* is the name of the AEM Forms Server and *[portnumber]* is the port number used to access the server.

1) In the WSDL URL field, type `http://`*[servername]*`:`*[portnumber]*`/soap/services/GeneratePDFServices?wsdl&lc_version=9.0.0&version=1.1&async=true`, and then click Load. After successfully loading the WSDL URL, in the Target URL box, the value http://*[server name]:[port number]*/soap/services/GeneratePDFServices is filled in for you.
2) Add the `?wsdl&async=true&lc&lc_version=9.0.0` to the provided URL. For example, change `http://`*[servername]*`:`*[portnumber]*`/soap/services/GeneratePDFServices` to `http://`*[servername]*`:`*[portnumber]*`/soap/services/GeneratePDFServices?wsdl&async=true&lc&lc_version=9.0.0`.
3) Select an asynchronous operation from the Operation list. For example, select HTMLToPDF to invoke the asynchronous operation to convert an HTML page to a PDF document.

## Invoking services in AEM Forms with document input and output values

You can invoke services that require file attachments as input values.

For example, service input and output variables of type document are represented in SOAP messages as file attachments. To provide input document values, the SOAP message includes a file attachment. Response messages include file attachments when web services have output variables of type document.

Files can be attached as MIME or MTOM attachments, base64-encoded text, or using the URL of the file. The SOAP message and the configuration of the Invoke Web Service operation are different for each type of attachment. If you are invoking a AEM Forms web service that returns a document value, you specify a method to attach the file to the SOAP response message.

**NOTE:** Web services that AEM Forms provides do not support MTOM attachments.

Before you configure file attachments, use the Web Service Options property of the Invoke Web Service operation to generate a template SOAP request message. (See AboutWeb Service Settings.)

**Configuring request messages**

The Web Service Settings dialog box automatically generates a SOAP message for invoking the web service. The message is based on the web service properties that you specified.

You add file attachments that represent document values of service input parameters to the SOAP message. The following XML code shows the general structure of a SOAP message that is used to a call a web service operation:

```
<soapenv:Envelope
xmlns:soapenv=http://schemas.xmlsoap.org/soap/envelope/
xmlns:ser="http://foo.net/services">
<soapenv:Header />
<soapenv:Body>
<ser:operation_name>
<ser:parameter_name>
<!-- elements for file attachments are placed here -->
</ser:parameter_name>
</ser:operation_name>
</soapenv:Body>
</soapenv:Envelope>
```

- The Body element contains one element that has the same name as the service operation.
- The operation_name element contains one element for each input parameter. The names of these elements are the same as the parameter names. For the web services of activated processes, the operation name is always invoke. Parameter names are the names of the input variables that are defined in the process.
- The method used to attach files to the SOAP message determines the elements that are used to identify file attachments. The methods available are MIME, MTOM, base64-encoded text, or file URLs. (Web services that AEM Forms provides do not support MTOM attachments.)

*MIME/MTOM attachment*

Use the Attachment tab of the Web Service Settings dialog box to specify the document value. Each row in the table represents a file attachment. The Part column contains the identification of the attachment. The web service determines the identification.

*Configure a MIME/MTOM attachment:*

1) On the Web Service Settings dialog box, click the Attachment tab.
2) To add rows to the table, click Load Attachment Part.
3) Note the values in the Part column. You use these values in step 7.
4) For each row in the table, click the Attachment field. Click the ellipsis button that appears to open XPath Builder.
5) For each row in the table, click the Content-Type cell and type the MIME type of the file that you are attaching. An example of a MIME type is`application/pdf`.

6) Click the Request tab, and add an attachmentID element inside each element that represents the parameter that takes a file attachment as the value.
7) Inside each attachmentID element, type the text that appears in the Part column for the attachment on the Attachment tab.

*Example*

The following SOAP request is an example of a configuration for a MIME or MTOM attachment. It is a request message for a process with an input parameter named documentIn, which is a document value.

```
<soapenv:Envelope
xmlns:soapenv=http://schemas.xmlsoap.org/soap/envelope/
xmlns:ser="http://adobe.com/idp/services">
<soapenv:Header/>
<soapenv:Body>
<ser:invoke>
<ser:documentIn>
<ser:attachmentID>
part0
</ser:attachmentID>
</ser:documentIn>
</ser:invoke>
</soapenv:Body>
</soapenv:Envelope>
```

*base64-encoded text attachment*

The content MIME type and the base64-encoded text representation of the file is included in the SOAP message. The content type is specified in a contentType element. The text is specified in a binaryData element.

To obtain the base64-encoded text, you can use the getDocContentBase64 XPath function.

The following procedure assumes that the base-64 encoded text that represents the document value is stored in a process variable.

*Configure a base64-encoded text attachment:*

1) On the Web Service Settings dialog box, click the Request tab.
2) Add a contentType element to the element that represents the parameter that specifies the document value.
3) Inside the contentType element, either type the content type or add an XPath expression that resolves to a string variable that contains the content type.
4) Add a binaryData element to the element that represents the parameter that takes the document value.
5) Inside the binaryData element, add an XPath expression that resolves to a string variable that contains the base64-encoded text that represents the document.

*Example*

The following SOAP request is an example of a configuration for a base64-encoded text attachment.It is a request message for a process with an input parameter named documentIn, which is a document value.

```
<soapenv:Envelope
xmlns:soapenv=http://schemas.xmlsoap.org/soap/envelope/
xmlns:ser="http://adobe.com/idp/services">
<soapenv:Header/>
<soapenv:Body>
<ser:invoke>
<ser:documentIn>
<ser:contentType>
{$/process_data/@contentType$}
</ser:contentType>
<ser:binaryData>
{$/process_data/@docBase64$}
</ser:binaryData>
</ser:documentIn>
</ser:invoke>
</soapenv:Body>
</soapenv:Envelope>
```

*File URL attachment*

The URL of the file is specified in a remoteURL element.

*Configure a file URL attachment:*

1) On the Web Service Settings dialog box, click the Request tab.
2) Inside the element that represents the parameter that takes the document value, add a remoteURL element.
3) Inside the remoteURL element, either type the URL of the file or add an XPath expression that resolves to a string variable that contains the URL.

*Example*

The following SOAP request is an example of a configuration for a file URL attachment. It is a request message for a process with an input parameter named documentIn, which is a document value.

```
<soapenv:Envelope
xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:ser="http://adobe.com/idp/services">
<soapenv:Header/>
<soapenv:Body>
<ser:invoke>
<ser:documentIn>
<ser:remoteURL>

</ser:remoteURL>
</ser:documentIn>
```

```
</ser:invoke>
</soapenv:Body>
</soapenv:Envelope>
```

### Configuring web service response messages

When a AEM Forms web service returns a document value, the value is an attachment to the SOAP response message. By default, AEM Forms web services store the result document value on the AEM forms Server. The web services also include the URL to the document in the response message. However, if the SOAP request message includes a MIME attachment, the response also uses a MIME attachment to return document values.

You can override the default behavior and specify the method that AEM Forms web services use to return document attachments. You can attach files to SOAP response messages as MIME attachments, base64-encoded text, or using the URL of the file. To specify the way files are attached to response messages, append one of the values in the following table to the Target URL. You specify the Target URL property on the Settings tab of the Web Service Settings dialog box:

| Attachment type | Text to append |
|---|---|
| base64-encoded text | ?blob=base64 |
| MIME | ?blob=mime |
| File URL | ?blob=http |

For example, the following value for Target URL causes the web service of a process to attach files to SOAP response messages as base64-encoded text. The process name is echoDocument:

```
http://localhost:8080/soap/services/echoDocument?blob=base64
```

*Override default SOAP response message attachment behavior:*

1) On the Web Service Settings dialog box, click the Settings tab.
2) In the Target URL box, append one of the following values to the URL:
   – ?blob=base64
   – ?blob=mime
   – ?blob=http

## Using Table data types for SAP web services

SAP web services can require a Table data type as an input or output parameter. An SAP table is similar to a database table composed of columns, where each row in the table represents a record.

AEM Forms uses a list of map values to represent an SAP Table:

- The list value represents the table.
- Each map value represents a row in the table.
- The map key is the name of the table column. The corresponding string value is the value for that column.

For example, an SAP table has the columns COL1, COL2, and COL3. The SAP web service requires two rows of table data as a value for the INPUT_TABLE input parameter. You create two map variables that contain string values: varMap1 and varMap2. You use the Set Value service to set the values in each map, using entries that are similar to the following expressions:

```
/process_data/varMap1[@id='COL1'] = 'value1'
/process_data/varMap1[@id='COL2'] = 'value2'
/process_data/varMap1[@id='COL3'] = 'value3'
/process_data/varMap2[@id='COL1'] = 'valueA'
/process_data/varMap2[@id='COL2'] = 'valueB'
/process_data/varMap2[@id='COL3'] = 'valueC'
```

You also create a list variable named SAPtable. You use the Set Value service to add the map values to the list:

```
/process_data/SAPtable=varMap1
/process_data/SAPtable=varMap2
```

**NOTE:** When no index is provided for the list, the values are appended after any existing items in the list.

The following XML represents the INPUT_TABLE parameter in the SOAP request:

```
<INPUT_TABLE)
{$ /process_data/SAPtable $}
</INPUT_TABLE>
```

For testing the SOAP request in the development environment, use an item element to represent a list item. Each item element contains one element for each table column:

- The element names are the column names.
- The element character data is the column value.

```
<INPUT_TABLE)
<item>
<COL1>value1</COL1>
<COL2>value2</COL2>
<COL3>value3</COL3>
</item>
<item>
<COL1>valueA</COL1>
<COL2>valueB</COL2>
<COL3>valueC</COL3>
</item>
</INPUT_TABLE>
```

# 22.55. XMP Utilities

Retrieves or replaces existing metadata in the PDF document. The metadata is specified as XML source in EXtensible Metadata Platform (XMP) format or as a reusable PDF document.

XMP provides a standard format for creating, processing, and interchanging metadata for a wide variety of applications. XMP uses the W3C standard Resource Description Framework (RDF).

In XMP, metadata consists of a set of properties that are associated with a document. Metadata includes the properties author, creator, keywords, producer, subject, and title of a PDF document. In a PDF document, metadata can be stored in two places:

- The file trailer dictionary of text, not in XML format using a `XMPUtilityMetadata` (`com.adobe.livecycle.xmputility.client.XMPUtilityMetadata`) value.
- In the Metadata dictionary of the document catalog. This dictionary contains metadata associated with the entire document. This information is represented in XMP format which can be saved to file.

For information about using the XMP Utilities service, see Services Reference for AEM Forms

.

## Export Metadata operation

Exports the metadata from the specified PDF document that you can save as process data.

The PDF document you specify can be stored as process data or specified directly as a file from the file system.

For information about the General and Route Evaluation property groups, see Commonoperation properties.

### Input properties

Property to specify the PDF document.

*Input PDF Document*

A `document` value that represents the PDF document.

If you provide a literal value, clicking the ellipsis button opens the Select Asset dialog box. (See About Select Asset.)

### Output properties

Property to specify the location to store the metadata.

*Output Metadata*

The location to store the metadata. The data type is `XMPUtilityMetadata`.

### Exceptions

This operation can throw an XMPUtilityException exception.

## Import Metadata operation

Imports metadata from process data and replaces the existing metadata in the specified PDF document.

For information about the General and Route Evaluation property groups, see Commonoperation properties.

**Input properties**

Properties to specify the new metadata and the PDF document.

*Input PDF Document*

A `document` value that represents the PDF document.

If you provide a literal value, clicking the ellipsis button opens the Select Asset dialog box. (See About Select Asset.)

*Metadata*

An `XMPUtilityMetadata` value containing the metadata to replace in the specified PDF document. Any null property values in the supplied `XMPUtilityMetadata` process data results in an empty attribute in the PDF document.

**Output properties**

Property to specify the location to store the PDF document containing new metadata.

*Output PDF*

The location to store the PDF document that contains new metadata. The data type is `document`.

**Exceptions**

This operation can throw an XMPUtilityException exception.

## Export XMP operation

Exports the metadata as XMP data from the specified PDF document. You can save the metadata as a PDF document to process data or a file to be reused.

For information about the General and Route Evaluation property groups, see Commonoperation properties.

**Input properties**

Property to specify the PDF document.

*Input PDF Document*

A `document` value that represents the PDF document.

If you provide a literal value, clicking the ellipsis button opens the Select Asset dialog box. (See About Select Asset.)

### Output properties

Property to specify the location to save the metadata.

*Output XMP*

The location to store the PDF document that contains new metadata. The data type is `document`.

### Exceptions

This operation can throw an XMPUtilityException exception.

## Import XMP operation

Imports metadata from a `document` value and replaces the existing metadata in the specified PDF document.

For information about the General and Route Evaluation property groups, see Commonoperation properties.

### Input properties

Properties to specify the PDF document and metadata.

*Input PDF Document*

A `document` value that represents the PDF document.

If you provide a literal value, clicking the ellipsis button opens the Select Asset dialog box. (See About Select Asset.)

*Input XMP Document*

A `document` value that represents the metadata formatted in XMP.

If you provide a literal value, clicking the ellipsis button opens the Select Asset dialog box. (See About Select Asset.)

### Output properties

Property to specify the location of the PDF document.

*Output PDF*

The location to save a PDF document that contains new metadata. The data type is `document`.

### Exceptions

This operation can throw an XMPUtilityException exception.

### XMP Utilities exceptions

The XMP Utilities service provides the following exception.

#### XMPUtilityException

Thrown when an error occurs importing metadata into a PDF document or exporting metadata from a PDF document.

## 22.56. XSLT Transformation

Enables processes to apply Extensible Stylesheet Language Transformations (XSLT) on XML documents.

For information about using the XSLT Transformation service, see Services Reference for AEM Forms

.

### XSLT Transformation service configuration

You can configure the XSLT Transformation service with the default Java class to use for performing the XSLT transformation. (See Editing service configurations.)

The service operation properties can override the value that you provide in the service configuration.

#### Factory Name

A `string` value that represents the fully qualified name of the Java class to use for performing XSLT transformations.

If no value is specified, the default factory configured in the Java Virtual Machine that runs the AEM forms Server is used. You need to specify a value if you have a specific reason to override the default factory.

### Transform operation

Transforms an XML document using an XSLT script that is stored as process data. The XML document is also stored as process data. You can save the resulting XML document as process data.

For information about the General and Route Evaluation property groups, see Commonoperation properties.

#### Factory Options properties

A property for specifying the XSLT engine to use.

*Factory Name*

A `string` value that represents the fully qualified name of the Java class to use for performing the XSLT transformation, such as `org.apache.xalan.xslt.XSLTProcessor`. The Java class used must

extend `javax.xml.transform.TransformerFactory` and must be available to the class loader of the AEM Forms Server.

**XML Source properties**

A property for specifying the source document.

*Source XML Document*

An `xml` value that represents the XML document on which the transformation is performed.

**XSLT Source properties**

A property for specifying the XSLT script to apply to the source document.

*XSLT*

A `string` value that represents the XSLT script.

To specify a literal value, click the ellipsis button to display the XSLT Source Text Input and Testing dialog box to use for specifying the value. (See AboutXSLT Source Text Input and Testing.)

**Transformation Result properties**

A property to use for saving the results of the transformation.

*Transformed XML*

The location to store the XML that has been transformed. The data type is `string`.

**Exceptions**

This operation can throw the following exceptions: Java.net.MalformedURLException, Java.io.IOException, Javax.xml.transform.TransformerException, Java.net.URISyntaxException, Javax.xml.parsers.ParserConfigurationException, Org.xml.sax.SAXException, Java.lang.ClassNotFoundException, Java.lang.InstantiationException, and Java.lang.IllegalAccessException.

## Transform using XSLT(URL) operation

Transforms an XML document using an XSLT script that is referenced using a URL. The XML document that is transformed is stored as process data. You can save the resulting XML document as process data.

For information about the General and Route Evaluation property groups, see Commonoperation properties.

**Factory Options properties**

A property for specifying the XSLT engine to use.

*Factory Name*

A `string` value that represents the fully qualified name of the Java class to use for performing the XSLT transformation, such as `org.apache.xalan.xslt.XSLTProcessor`. The Java class used must extend `javax.xml.transform.TransformerFactory` and must be available to the class loader of the AEM Forms Server.

**XML Source properties**

A property for specifying the source document.

*Source XML Document*

An `xml` value that represents the XML document on which the transformation is performed.

**XSLT Source properties**

A property for specifying the XSLT script to apply to the source document.

*XSLT URL*

A `string` value that represents the URL of the XSLT script.

To test the script, select literal value, and then click the ellipsis button to display the XSLT Source Text Input and Testing dialog box to use for specifying the value. (See AboutXSLT Source Text Input and Testing.)

**Transformation Result properties**

A property to use for saving the results of the transformation.

*Transformed XML*

The location to store the result (formatted as XML) that has been transformed. The data type is `string`.

**Exceptions**

This operation can throw the following exceptions: Java.net.MalformedURLException, Java.io.IOException, Javax.xml.transform.TransformerException, Java.net.URISyntaxException, Javax.xml.parsers.ParserConfigurationException, Org.xml.sax.SAXException, Java.lang.ClassNotFoundException, Java.lang.InstantiationException, and Java.lang.IllegalAccessException.

## XSLT Transformation exceptions

The XLST Transformation service provides the following exceptions for throwing exception events.

**Java.lang.IllegalAccessException**

Thrown when an operation attempts to perform an operation or access data that is not allowed.

### Java.net.MalformedURLException

Thrown when a URL passed to an object is not in a valid format when an operation runs.

### Java.io.IOException

Thrown to indicate that an I/O problem of some sort occurred when an operation runs.

### Javax.xml.transform.TransformerException

Thrown when a serious error occurred due to an internal configuration when an operation runs.

### Java.net.URISyntaxException

Thrown when a string value cannot be parsed as a URI when an operation runs.

### Javax.xml.parsers.ParserConfigurationException

Thrown when an error occurred while configuring the XML parser when an operation runs.

### Org.xml.sax.SAXException

Thrown when a generic SAX error or warning occurs when an operation runs.

### Java.lang.ClassNotFoundException

Thrown when a service or operation cannot be found.

### Java.lang.InstantiationException

Thrown when an attempt is made to use an operation or service that is not meant to be run.

### Java.lang.IllegalAccessException

Thrown when an operation attempts to perform an operation or access data that is not allowed.

## About XSLT Source Text Input and Testing

The XSLT Source Text Input and Testing dialog box lets you specify the XSLT script to use for transforming the XML source. You can also test the transformation and see the result.

### XSLT

Use this tab to specify the XSLT script to use to transform the XML source. The text that you provide is also used for testing purposes.

You can type directly in the text editing area, or paste the script (right-click the editing area and click Paste).

### Test XML

Use this tab to provide the XML code to use for testing the XSLT script that you provided on the XSLT tab.

You can type directly in the text editing area, or paste the script (right-click the editing area and click Paste).

### Test Result

Use this tab to apply the XSLT script to the test XML.

Click Test to transform the test XML. The resulting XML appears on the Text tab. The rendered HTML appears on the HTML tab.

To clear the test results, click Clear.

# List of Figures

# List of Tables

# Index

## A

abstract modelling elements:about 172
abstract modelling elements:adding 173
abstract modelling elements:replacing 173
access control lists, Assign Task operations 257
access permissions:about 65
access permissions:creating 66
access permissions:removing 68
access permissions:viewing 66
Acrobat PDF forms 216
AcrobatVersion variable type 507
AcrobatVersion-OutputService variable type 508
activating:event types 406
activating:services 70
activity elements:usage 79
ActivityGuideRenderSpec (8.2) variable type 547
Add Invisible Signature Field operation:Signature service 1156
Add Signature Validation Information operation:Signature service 1162
Add Subject DN dialog box 1247
Add Users and Groups dialog box 1144
Add Visible Signature Field operation:Signature service 1170
add-days-to-date XPath function 305
add-days-to-dateTime XPath function 305
add-hours-to-dateTime XPath function 306
adding an application 39
adding:abstract modelling elements 173
adding:annotation elements 133
adding:branches 185
adding:conditions, to routes 181
adding:event catches 196
adding:event throws, receives, and start points 192
adding:gateway elements 183
adding:operations 167
adding:routes 178
add-minutes-to-dateTime XPath function 306
add-months-to-date XPath function 307
add-months-to-dateTime XPath function 307
add-seconds-to-dateTime XPath function 308
add-years-to-date XPath function 308
add-years-to-dateTime XPath function 309
Adobe AEM Forms Workspace ES:TaskManager endpoints 386
Adobe PDF forms 216
Adobe XML forms 216
AEM Forms server:application server logs 398
AEM Forms server:configuring for proxy servers 764
AEM Forms Workbench ES:configuring for proxy servers 764
AEM Forms Workspace ES:best practices 266
AEM Forms Workspace ES:card instructions 269
AEM Forms Workspace ES:card titles 268
AEM Forms Workspace ES:category names 266
AEM Forms Workspace ES:configuring attachments and notes 259
AEM Forms Workspace ES:configuring deadlines 248
AEM Forms Workspace ES:configuring delegations and consultations 254
AEM Forms Workspace ES:configuring escalations 249

AEM Forms Workspace ES:configuring reminders 247
AEM Forms Workspace ES:configuring task priority 260
AEM Forms Workspace ES:Enduser UI Items variable property group 507
AEM Forms Workspace ES:process cards 267
AEM Forms Workspace ES:process data 166
AEM Forms Workspace ES:task completion options 238
AND-WAIT gateway control type 85, 89, 183
annotation elements, adding 133
application server logs 398
Applications, about 35
Applications:adding 39
Applications:assets 41
Applications:deploying 48
Applications:removing 39
Apply policy operation:Rights Management service 1128
Apply Usage Rights operation:Acrobat Reader DC extensions service 1118
archive files:See LCA files
Archive wiarzd:archival methods 207
Archive wizard 206
Archive wizard: e-mail address settings 210
Archive wizard:document selection 207
Archive wizard:e-mail server settings 209
Archive wizard:file system settings 210
Archive wizard:printer settings 211
Archive wizard:summary 211
archives:creating asynchronously 53
Assembler DDX Editor dialog box 770
Assembler service 769
Assembler service:Quick Start 101
AssemblerOptionSpec variable type 512
AssemblerResult variable type 514
assets 41
Assign Task operation:task instructions 237
Assign Task operation:User service 1263
Assign Task operations:access control lists 257
Assign Task operations:attachments and notes 259
Assign Task operations:delegations and consultations 254
Assign Task operations:specifying the user 229
assigning tasks:to groups 232
assigning tasks:to specific users 232
assigning tasks:using XPath expressions 233
asynchronous branches 185
asynchronous event types 408
Attachments dialog box 934
attachments:configuring 259
attachments:document variable attributes 264
attachments:related event types 409
attributes, document variable data 264

## B

barcode data, in processes 110
Barcoded Forms service 778
binary variable type 516
boolean variable type 516
boolean XPath function 299
branches: types and operations 187
branches:about 185
branches:adding to gateways 185
branches:parallel 85
branches:stalled 188, 395
branches:transactions 188
Business Calendar Date variable type 517
business calendars 247
businessCalendarWait operation:Wait Point service 1290
byte variable type 517

## C

Call Stored Procedure operation:JDBC service 1049
Callable Statement Info Editor dialog box 1053
catching events 195
categories:event groups 405
categories:event types 405
categories:events 191
category names:AEM forms Workspace ES 266
ceiling XPath function 332
CentralResult variable type 518
Certificate Encrypt PDF operation:Encryption service 936
Certificate Encryption Options Spec variable type 519
Certify PDF operation:Signature service 1175
character classes:regular expressions 976

Character Set (barcoded forms) variable type 522
Character Set (File Utilities) variable type 523
Clear Signature Field operation:Signature service 1188
Clone PDF operation 1106
closing process 135
coercion of data 169
collections:list variable type 571
collections:map variable type 574
collections:XPath functions 300
common properties, variable types 505
completion of processes 92
completion of tasks:forced 244
Components view:activating services 70
Components view:configuring services 71
Components view:deactivating services 72
Components view:installing compononents 69
Components view:opening 69
Components view:patching compoonents 69
Components view:removing services 71
Components view:starting components 70
Components view:stopping components and services 71
Components view:uninstalling components 72
concat XPath function 334
conditions:about 86, 177, 181
conditions:adding 181
configuration parameters:about 162
configuring:Email service 919
configuring:FTP service 1023
configuring:JDBC service 1049
configuring:LDAP service 1065
configuring:services 71
configuring:XSLT Transformation service 1308
connections, logging in 31
consultations, configuring 254
contains XPath function 334
Content Repository Connector for EMC Documentum service 803
Content Repository Connector for IBM Content Manager service 828
Content Repository Connector for IBM DB2 Content Manager:about 828
Content Repository Connector for IBM FileNet service 856
Content Repository Connector service:EMC Documentum 803
Content Repository Connector service:IBM FileNet connector 856
Convert PDF service 899
Convert PDF service:about 899
Convert PDF to XDP operation:PDF Utilities service 1106
Convert XDP to PDF operation:PDF Utilities service 1107
converting data types 294
copying:branches 185
copying:process diagram elements 132
count XPath function 331
counters, loops 91
Create folder operation:IBM Content Manager connector 832
Create Folders operation:EMC Documentum connector 803
Create Folders operation:IBM FileNet connector 857
Create Folders with type and attributes operation:EMC Documentum connector 805
Create item operation:IBM Content Manager connector 828
Create policy from template operation:Rights Management service 1132
Create Relationship operation:EMC Documentum connector 808
Create Relationship operation:IBM Content Manager connector 834
Create Relationship operation:IBM FileNet connector 858
create_time, process property 77
CreatePDF2 operation:Distiller service 911
CreatePDF2 operation:Generate PDF service 1040
creating:custom event types 401
creating:processes 126
creating:processes, by importing 139
creating:processes, Quick Starts 96
creating:variables 163
creator_id, process property 77
Credential variable type 527
CRLOptionSpec variable type 529
CSS:relationships 63
current-date XPath function 309
current-dateTime XPath function 310

current-time XPath function 310
custom event types 401
custom event types:event data and messages 403
custom event types:event data templates 404
custom event types:messages templates 404
customizing:record and playback behavior 391
customizing:validation behavior 363

## D

data coercion 169
data collections 295
data maps, creating 204
data models, processes 77
data type conversion 294
database, variables 80
datatype specific settings:Document Form variable type 536
datatype specific settings:Form variable type 543
datatype specific settings:xfaForm variable type 757
date and time:as parameter values 340
date and time:XPath functions 305
date variable type 532
date-equal XPath function 310
date-greater-than XPath function 311
date-less-than XPath function 311
dateTime variable type 533
dateTime-equal XPath function 312
dateTime-greater-than XPath function 312
dateTime-less-than XPath function 313
deactivating:event types 406
deactivating:services 72
deadlines, configuring for tasks 248
debugging process versions:using server logs 398
decimal variable type 533
Decision Point service 908
Decision Point service:about 908
decisions, routes 86
Decode operation:barcoded forms service 778
Default Render ES Update 1 service 908
Default Render service:about 908
defining abstract modelling elements 173
delegations, configuring 254
Delete Content operation:EMC Documentum connector 810
Delete file operation:FTP service 1023
Delete operation:File Utilities service 958
deleting an application 39
deleting recordings 390
deleting:operations 167
deleting:process variables 166
deleting:routes 178
Delimiter variable type 534
deploying applications 48
deploying:process 138
deserialize XPath function 330
Designer Help:accessing 60
directly invoking process versions 385
Dispatch Workflow Step operation:IBM FileNet connector 1112
Distiller service 910
Distiller service:about 910
DocConverter service:about 913
DocInfo variable type 535
Document Form variable type 535
Document Form variable type:datatype specific settings 536
Document Form variable type:render service 536
Document Form variable type:submit service 536
document variable type 534
document variables:data attributes 264
document XPath functions 326
double variable type 538

## E

editing:process variables 165
EJB connections 31
Email service 918
Email service:configuring 919
email start point 130
Email Template Editor 1278
email, invoking processes 386
email:for task notifications 250
empty-list XPath function 300
empty-map XPath function 301
Encryption Identity variable type 538
Encryption service 935
Encryption service:about 935
EncryptionTypeResult variable type 540
ends-with XPath function 335
Enduser UI Items, about 507
errors, stalled operations and branches 395

escalations, configuring for tasks 249
event catches:about 195
event catches:adding 196
event catches:debugging process versions 397
event categories:about 191, 405
event categories:displaying 405
event data templates, custom event types 403, 404
event groups:about 405
event groups:displaying 405
event receives:about 194
event receives:adding 192
event start points:about 198
event start points:adding 192
event throws:adding 192
event throws:start points 90
event types:about 190
event types:about throwing 192
event types:activating and deactivating 406
event types:asynchronous 408
event types:categories 405
event types:creating custom types 401
event types:Exception 444
event types:importing and exporting 405
event types:Rights Management service 408
event types:task attachments 409
event types:task management 409
events:about 190
events:as start activity 200
events:data 190
events:data maps 204
events:filters 202
events:multi-choice example 88
events:Quick Start 107
events:synchronizing merge 89
Exception event type 444
exception events, and operations 198
exceptions:Acrobat Reader DC extensions service 1122
exceptions:Assembler service 776
exceptions:barcoded forms service 781
exceptions:Content Repository Connector for EMC Documentum service 828
exceptions:Convert PDF service 907
exceptions:Distiller service 912
exceptions:DocConverter service 918
exceptions:Email service 933
exceptions:Encryption service 944
exceptions:File Utilities service 977
exceptions:Form Augmenter service 984
exceptions:Form Data Integration service 987
exceptions:Forms service 1022
exceptions:FTP service 1036
exceptions:Generate PDF service 1048
exceptions:IBM Content Manager connector 856
exceptions:IBM FileNet connector 876
exceptions:Output service 1103
exceptions:PDF Utilities service 1112
exceptions:Rights Management service 1144
exceptions:Signature service 1245
exceptions:User service 1274
exceptions:Variable Logger service 1290
exceptions:Web Service service 1293
exceptions:XMP Utilities service 1308
exceptions:XSLT Transformation service 1310
Execute DQL Query operation:EMC Documentum connector 812
execute operation:Decision Point service 908
execute operation:Set Value service 1147
execute operation:Stall service 1249
Execute Script service 945
Execute SQL Statement operation:JDBC service 1050
executeScript operation:Execute Script service 945
executioon errors 395
Exists operation:File Utilities service 959
Export Metadata operation:XMP Utilities service 1305
Export XMP operation:XMP Utilities service 1306
exporting:event types 405
exporting:process 139
ExportPDF2 operation:Generate PDF service 1042
expressions:operators in 339
expressions:routing conditions 181
Extract to XML operation:barcoded forms service 780

## F

false XPath function 299
FieldMDPOptionSpec variable type 540
File Utilities service 958

File Utilities service:about 958
File Utilities service:regular expressions 975
filters:events 202
filters:Resources view 62
Find operation:File Utilities service 959
Find Type variable type 763
Flex applications 216
float variable type 542
floor XPath function 333
forcing task completion 244
Form Augmenter service 977
Form Augmenter service:about 977
Form Data Integration service 985
Form Data Integration service:about 985
form data, in the process data model 218, 262
form data:adding nodes 265
form data:changing 292
form data:in XPath Expressions 291
form data:multiple forms 99
form designs:creating 56
form designs:saving 60
form fields, in XPath Builder 219
form schema:adding nodes 265
form schema:in the process data model 218, 262
Form variable type 542
Form variable type:datatype specific settings 543
Form variable type:render service 543
Form variable type:schema settings 544
Form variable type:submit service 543
format-date XPath function 323
format-dateTime XPath function 324
FormDatatype variable type 545
FormGuideRenderSpec variable type 547
FormModel variable type 545
FormPreference variable type 545
forms:shared data 99
forms:submit options 238
forms:types 216
FormsResult variable type 551
forwarded tasks, sharing 255
fragments:relationships 63
FTP FileInfo variable type 554
FTP service 1022
FTP service:about 1022
FTP service:configuring 1023
functions, in XPath Builder 285
functions:data type conversion 294

## G

gateway branches:adding operations 186
gateway elements:about 183
gateway elements:adding 183
gateway elements:adding branches 185
gateway elements:multi-choice example 87
gateway elements:parallel branches 85
gateway elements:synchronizing merge 89
General properties of operations 766
General variable property group, about 506
generatePDFOutput operation:Output service 1079
generatePrintedOutput operation:Output service 1089
Get Certifying Signature Field operation:Signature service 1189
Get Credential Usage Rights operation:Reader Extensions service 1121
Get Document Usage Rights operation:Reader Extensions service 1120
Get Legal Attestation operation:Signature service 1190
Get license identifier operation:Rights Management service 1135
Get linked LC Assets Location operation:EMC Documentum connector 814
Get list of files operation:FTP service 1026
Get multiple documents operation:FTP service 1027
Get operation:FTP service 1024
Get PDF Encryption Type operation:Encryption service 938
Get PDF Properties operation:PDF Utilities service 1108
Get PDF Save Mode operation:PDF Utilities service 1109
Get Related Items operation:IBM Content Manager connector 842
Get Related operation:EMC Documentum connector 816
Get Related operation:IBM FileNet connector 864
Get Signature Field List operation:Signature service 1192
Get Signature operation:Signature service 1190

Get Signed Version operation:Signature service 1192
Get to file system operation:FTP service 1029
Get value of a form field operation:Form Augmenter service 978
get-collection-size XPath function 302
get-day-from-date XPath function 313
get-days-from-date-difference XPath function 314
get-days-from-dateTime-difference XPath function 314
getDocAttribute XPath function 327
getDocContent XPath function 328
getDocLength XPath function 326
get-hours-from-dateTime XPath function 315
get-hours-from-dateTime-difference XPath function 315
get-hours-from-time XPath function 316
get-map-keys XPath function 302
get-minutes-from-dateTime XPath function 316
get-minutes-from-dateTime-difference XPath function 316
get-minutes-from-time XPath function 317
get-month-from-date XPath function 317
get-month-from-dateTime XPath function 318
get-months-from-date-difference XPath function 318
get-seconds-from-dateTime XPath function 319
get-seconds-from-dateTime-difference XPath function 319
get-seconds-from-time XPath function 320
get-timezone-from-date XPath function 320
get-timezone-from-dateTime XPath function 321
get-timezone-from-time XPath function 321
get-year-from-date XPath function 321
get-year-from-dateTime XPath function 322
get-years-from-date-difference XPath function 322
get-years-from-dateTime-difference XPath function 322
groups, assigning tasks 232

## H

HashAlgorithm variable type 557
HtmlFileToPDF operation:Generate PDF service 1044
HTMLRenderSpec variable type 558
HTMLToolbar variable type 565
HtmlToPDF operation:Generate PDF service 1045
human-centric processes:about 212
human-centric processes:involving users 229

## I

id, process property 77
IdentityDetails variable type 566
IdentityInformation variable type 567
IdentityStatus variable type 568
IGetLinkedLCAssetsLocationResult variable type 568
ILoginSettings.LoginMode variable type 569
images, for processes 126
images:relationships 63
implicit variable properties 505
import Data operation:Form Data Integration service 987
Import Metadata operation:XMP Utilities service 1305
Import XMP operation:XMP Utilities service 1307
importing:event types 405
importing:process versions 139
Inject Form Bridge operation:Form Augmenter service 979
input data, processes 77
Insert Workflow XFO Data operation:Form Augmenter service 980
installing components 69
instructions, for tasks 237
instructions, process cards 269
int variable type 570
Invocation Policy:subprocesses 768
invoke operation, process services 767
Invoke operation:Default Render service 909
Invoke operation:Print PDF Package service 1117
Invoke operation:Render Form Guide service 1123
Invoke operation:Render HTML Form service 1124
Invoke operation:Render PDF Form service 1126
Invoke operation:Submit Form Guide service 1250
Invoke operation:Submit HTML Form service 1251
Invoke operation:Submit PDF Form service 1253
Invoke Web Service operation:Web Service service 1292
invokeDDX operation:Assembler service 770
invokeOneDocument operation:Assembler ser-

vice 774
invoking process versions:about 385
invoking process versions:directly 385
invoking processes 386
Is Absolute operation:File Utilities service 961
Is Directory operation:File Utilities service 961
Is File operation:File Utilities service 962
Is Hidden operation:File Utilities service 962
is-null XPath function 330
isPDFA operation:DocConverter service 913, 915
iteration in processes 91

## J

JDBC service 1048
JDBC service:about 1048
JDBC service:configuring 1049

## L

labels, routes 180
last XPath function 331
lc7form variable type 570
LCA files:about 50
LCCPLM service 1059
LCCPLM service:about 1059
LDAP Query operation:LDAP service 1066
LDAP Query Options Editor dialog box 1070
LDAP Query to XML operation:LDAP service 1068
LDAP service 1063
LDAP service:about 1063
LDAP service:assigning tasks 96
LDAP service:configuring 1065
list variable type 571
list variables:indexes 297
list variables:XPath expressions 295
literal characters:regular expressions 975
literal values 168
log operation: debugging process versions 396
log operation:Variable Logger service 1288
logging in 30
logging out 33
LoginMode variable type 572
long variable type 573
long-lived processes:about 81
Lookup and Insert Workflow XFO Data operation:Form Augmenter service 982
loops, in process diagrams 91
lower-case XPath function 335

## M

magnification, process diagrams 132
Make Directory operation:File Utilities service 963
Make Directory Recursively operation:File Utilities service 964
map variable type 574
map variables:keys 297
map variables:XPath expresssions 295
MDPPermissions variable type 574
Message Body Editor dialog box 933
message data, events 190
message templates, custom event types 403, 404
MessageFormatEnum variable type 574
messages:SOAP invocation 1297
miscellaneous XPath functions 330
Modify Signature operation:Signature service 1194
modifying process 136
moving:views 72
Multiple Clone of PDF operation:PDF Utilities service 1109

## N

names of operations 766
node set XPath functions 331
node sets 293
normalize-space XPath function 335
not XPath function 299
notes:configuring 259
notes:document variable attributes 264
notifications:email templates 251
notifications:for tasks 250
NO-WAIT gateway control type 183
number functions 332
number XPath function 333

## O

object variable type 575
OCSPOptionSpec variable type 575
opening process 135
opening:process version recordings 386
operation names 766
operation properties:All view 769
operation properties:Basic view 769

operation property values, template 171
operations:about 167
operations:adding and deleting 167
operations:adding to gateways 186
operations:attaching events 196
operations:branch types 187
operations:common properties 766
operations:configuring 169
operations:exceptions 198
operations:input and output data 168
operations:long-lived 188
operations:stalled 395
operations:transaction-aware 188
operators, in XPath Builder 288
operators, XPath expressions 339
OR-WAIT gateway control type 183
Out Of Office 230
output data, processes 77
Output service 1078
OutputJog variable type 580
OutputResult variable type 581
OutputType variable type 582

## P

Pagination variable type 583
parallel branches:merging 85, 90
parallel branches:process modelling example 85
parse-date XPath function 324
parse-dateTime XPath function 325
Password Encrypt PDF operation:Encryption service 939
Password Encryption Type variable type 588
PasswordEncryptionOptionSpec variable type 585
patching components 69
patExecContext object methods 946
PathValidationOptionSpec variable type 589
PDF Utilities service 1105
PDFAConformance variable type 591
PDFAConversionOptionSpec variable type 592
PDFAConversionResult variable type 595
PDFARevisionNumber variable type 596
PDFAValidationOptionSpec variable type 597
PDFAValidationResult variable type 599
PDFDcoumentVerificationInfo variable type 600
PDFFormRenderSpec variable type 601
PDFLegalWarnings variable type 609
PDFOutputOptionsSpec variable type 614
PDFPropertiesOptionSpec variable type 620
PDFPropertiesResult variable type 621
PDFSeedValueOptionSpec variable type 623
PDFSignature variable type 634
PDFSignatureAppearanceOptionSpec variable type 635
PDFSignatureField variable type 639
PDFSignatureFieldProperties variable type 640
PDFSignatureVerificationInfo variable type 648
PDFSignatureVerificationResult variable type 650
PDFUtilitySaveMode variable type 653
permissions for Workbench ES users 33
perspectives:Form Design 54
perspectives:Process Design 74
perspectives:restoring default layout 73
perspectives:saving 73
playback mode, process version recordings 386
playing back process version recordings 386
playing recordings 388
pools:about 174
pools:naming 176
ports, EJB connections 31
position XPath function 332
PositionRectangle variable type 656
preserving process results 206
Principal Reference variable type 657
Print PDF Package service 1117
Print PDF Package service:about 1112, 1117
PrintedOutputOptionsSpec variable type 659
PrinterProtocol variable type 666
PrintFormat variable type 667
process cards:AEM forms Workspace ES 267
process cards:instructions 269
process cards:titles 268
process data model:about 77
process data model:form data 218, 262
process data model:node sets 293
process data, creating variables 163
process data:converting data types 294
process data:form field values 291
Process Design perspective 74
process diagrams:about 76
process diagrams:annotations 133
process diagrams:zooming 132
Process Engine Connector for IBM FileNet service

1112
Process Engine Connector service:IBM FileNet connector 1112
process instances 80
process variables:about 162
process variables:deleting 166
process variables:editing 165
process versions:debugging using event catches 397
process versions:debugging using server logs 398
process versions:debugging using the Stall service 398
process versions:debugging using the Variable Logger service 396
process versions:importing 139
process versions:invoking 385, 386
process versions:LCA files 50
process versions:run time 80
process versions:validation messages 362, 364
process wizard 126
process:deploying and undeploying 138
process:Email start point 130
process:exporting 139
process:modifying 136
process:opening and closing 135
process:preserving results 206
process:Watched Folder start point 131
process:Workspace start point 128, 129
processes: short- and long-lived 81
processes:completion 92
processes:creating 126
processes:human-centric 212
processes:invoking 386
processes:involving users 229
processes:process services reference 767
processes:start activities 177
processes:test cases 394
processFormSubmission operation:Forms service 989
properties common to operations 766
proxy servers:using 764
PSLevel variable type 670
Put from file system operation:FTP service 1033
Put multiple documents operation:FTP service 1035
Put operation:FTP service 1031

## Q

quantifiers:regular expressions 976
Query for Multiple Rows as XML operation:JDBC service 1051
Query SIngle Row operation:JDBC service 1051
Quick Starts:Assembler service 101
Quick Starts:barcoded form data 110
Quick Starts:creating processes 96
Quick Starts:events 107
Quick Starts:form data 99
Quick Starts:Rights Management service 104
Quick Starts:role-based task assignment 96

## R

Read Document operation:File Utilities service 964
Read Resource Content operation:Repository service 1127
Read String operation:File Utilities service 965
Read XML operation:File Utilities service 966
ReaderExtensionsOptionSpec variable type 671
Reason variable type 674
Receive operation:Email service 920
ReceiveProtocolEnum variable type 674
receiving events 194
recording and playing back:customizing behavior 391
recording and playing back:deleting recordings 390
recording and playing back:playback mode 386
recording and playing back:playing 388
recording and playing back:playing back recordings 386
recording and playing back:stopping and starting 384
regular expressions:character classes 976
regular expressions:literal characters 975
regular expressions:quantifiers 976
regular expressions:syntax 975
RelationInfo data type 675
relationship, resources 63
reminders, configuring for tasks 247
Remove Data Fields operation:Form Augmenter service 984
Remove PDF Certificate Encryption operation:Encryption service 941

Remove PDF Password Encryption operation:Encryption service 942
Remove policy security operation:Rights Management service 1139
Remove Signature Field operation:Signature service 1199
Remove Usage Rights operation:Acrobat Reader DC extensions service 1121
removing an application 39
removing services 71
Rename To operation:File Utilities service 967
Render Form Guide ES Update 1 service 1122
Render Form Guide service:about 1122
Render HTML Form ES Update 1 service 1124
Render HTML Form service:about 1124
Render PDF Form ES Update 1 service 1125
Render PDF Form service:about 1125
render service:Document Form variable type 536
render service:Form variable type 543
render service:xfaForm variable type 758
RenderAtClient variable type 676
renderHTMLForm operation:Forms service 999
renderHTMLForm2 operation:Forms service 991
RenderOptionsSpec-FormsService variable type 677
RenderOptionsSpec-OutputServer variable type 692
renderPDFForm operation:Forms service 1015
renderPDFForm2 operation:Forms service 1008
Repository service 1127
Repository service:about 1127
Resources view:about 62
Resources view:access permissions 65
Resources view:access permissions, creating 66
Resources view:access permissions, removing 68
Resources view:access permissions, viewing 66
Resources view:filters 62
Resources view:using older resource versions 65
Resources view:version history 64
Resources view:versions 64
resources:access permissions 65
resources:access permissions, creating 66
resources:access permissions, removing 68
resources:relationships 63
resources:using older versions 65
resources:viewing access permissions 66
restoring perspectives 73
RetainSignatureField variable type 697
Retrieve Content operation:EMC Documentum connector 818
Retrieve Content operation:IBM FileNet connector 866
Retrieve Item operation:IBM Content Manager connector 844
Retrieve Workflow Step Parameters operation:IBM FileNet connector 1113
RevocationCheckStyle variable type 704
Revoke license operation:Rights Management service 1140
Rights Management service 1128
Rights Management service:about 1128
Rights Management service:event types 408
Rights Management service:Quick Start 104
RMAdminEvent event type 410
RMDocumentEvent event type 411
RMErrorEvent event type 412
RMPolicyEvent event type 414
RMPolicySetEvent event type 415
RMServerEvent event type 416
RMUserEvent event type 418
roles, assigning tasks 96
round XPath function 333
route evaluation 182
Route Evaluation properties of operations 767
route names, as form submit options 238
routes:about 177
routes:adding 178
routes:conditions 177, 181
routes:labels 180
routes:modifying the shape 180
routes:order of evaluation 182
Rule variable type 710

## S

saving form designs 60
saving perspectives 73
scheduleWait operation:Wait Point service 1291
schema settings, Form variable type 544
schema settings, xfaForm variable type 760
schemas:relationships 63
search filters:creating 1074
search filters:syntax 1076

Search Items operation:IBM Content Manager connector 847
Search Query Builder dialog box 1074
Select Group dialog box 1277
Select User dialog box 1275
selecting assets 769
Send With Document operation:Email service 924
Send With Map of Attachments operation:Email service 928
sent to printer:Output service 1099
serialize XPath function 331
servers:connections 31
servers:logging in 30
servers:logs 398
service configuration:EMC Documentum connector 803
service configuration:IBM FileNet connector 856
services:about 75
services:activating 70
Services:Assembler 769
Services:barcoded forms 778
services:connecting through proxy servers 764
Services:Content Repository Connector for EMC Documentum 803
Services:Content Repository Connector for IBM Content Manager 828
Services:Content Repository Connector for IBM FileNet 856
Services:Convert PDF 899
services:deactivating 72
Services:Decision Point 908
Services:Default Render ES Update 1 908
Services:Distiller 910
services:editing service configurations 71
Services:Email 918
Services:Encryption 935
Services:Execute Script 945
Services:File Utilities 958
services:for processes 767
Services:Form Augmenter 977
Services:Form Data Integration 985
Services:FTP 1022
Services:JDBC 1048
Services:LCCPLM 1059
Services:LDAP 1063
Services:Output 1078
Services:PDF Utilities 1105
Services:Print PDF Package 1117
Services:Process Engine Connector for IBM FileNet 1112
services:removing 71
Services:Render Form Guide ES Update 1 1122
Services:Render HTML Form ES Update 1 1124
Services:Render PDF Form ES Update 1 1125
Services:Repository 1127
Services:Rights Management 1128
Services:Set Value 1146
Services:Signature 1147
Services:Stall 1249
services:starting 70
services:stopping 71
Services:Submit Form Guide ES Update 1 1250
Services:Submit HTML Form ES Update 1 1251
Services:Submit PDF Form ES Update 1 1253
Services:User 1255
Services:User Lookup 1280
Services:Variable Logger 1288
Services:Wait Point 1290
Services:Web Service 1291
Services:XMP Utilities 1304
Services:XSLT Tranformation 1308
Set Link To LC Assets operation:EMC Documentum connector 822
Set Link To LC Assets operation:IBM Content Manager connector 849
Set Link to LC Assets operation:IBM FileNet connector 870
Set PDF Save Mode operation:PDF Utilities service 1110
Set Read Only operation:File Utilities service 968
Set Value service 1146
Set Value service:about 1146
Set Value service:form data 99
Set Workflow Step Parameters operation:IBM FileNet connector 1115
setDocAttribute XPath function 327
setDocContent XPath function 328
shape, routes 180
short variable type 710
short-lived processes:about 81
Sign Signature Field operation:Signature service 1200

Signature service 1147
Signature service:about 1147
Signature Service:Quick Starts 104
SignatureProperties variable type 711
SOAP invocation messages 1297
special characters:quantifiers 976
special characters:XPath functions 298
SPI Properties dialog box 1246
SQL Statement Info Editor dialog box 1055
SQL statements:executing 1055
SQL statements:including parameters 1056
SQL statements:including XPath expressions 1055
Stall service 1249
Stall service:about 1249
Stall service:debugging process versions 398
stalled branches 188
stalled operations and branches 395
Staple variable type 712
start activities:events 200
start activities:specifying 177
start points:about 198
start points:example 88
starting components and services 70
StartReviewStage event type 419
starts-with XPath function 335
status, process property 77
stopping and starting recording 384
stopping components and services 71
Store Content operation:EMC Documentum connector 824
Store Content operation:IBM FileNet connector 872
stored procedures:calling 1053
stored procedures:calls including XPath expressions 1054
stored procedures:defining parameters 1054
string variable type 713
string XPath function 336
string XPath functions 334
string-length XPath function 338
Submit Form Guide ES Update 1 service 1250
Submit Form Guide service:about 1250
Submit HTML Form ES Update 1 service 1251
Submit HTML Form service:about 1251
Submit PDF Form ES Update 1 service 1253
Submit PDF Form service:about 1253
submit service:Document Form variable type 536
submit service:Form variable type 543
submit service:xfaForm variable type 758
subprocesses:design principle 79
subprocesses:events 88
subprocesses:Invocation Policy 768
subprocesses:invoke operation 767
substring XPath function 337
substring-after XPath function 336
substring-before XPath function 337
sum XPath function 332
swimlanes:about 174
swimlanes:modifying 175
Switch policy operation:Rights Management service 1141
synchronous branches 185

## T

task completion:forced 244
Task Date variable type 719
Task Deadline variable type 720
Task Delegate and Consult variable type 722
Task Priority variable type 723
Task Reminder variable type 724
Task Routes and Priority variable type 728
Task User Selection variable type 730
TaskAttachmentAdded event type 420
TaskAttachmentDeleted event type 421
TaskAttachmentInfoUpdated event type 422
TaskAttachmentUpdated event type 424
TaskCancelled event type 425
TaskClaimed event type 426
TaskCompleted event type 427
TaskCompletedWithData event type 428
TaskConsulted event type 429
TaskCreated event type 430
TaskDeadlined event type 431
TaskEscalated event type 432
TaskFormDataSaved event type 433
TaskForwarded event type 434
TaskLocked event type 435
TaskManager, endpoints 386
TaskReassigned event type 436
TaskRejected event type 437
TaskReminderSent event type 438
tasks:assigning based on roles 96

tasks:assinging to groups 232
tasks:assinging to specific users 232
tasks:assinging using XPath expressions 233
tasks:deadlines 248
tasks:email templates for notifications 251
tasks:escalations 249
tasks:instructions 237
tasks:notifications 250
tasks:priority 260
tasks:reassigning when out of office 230
tasks:related event types 409
tasks:reminders 247
tasks:sharing forwarded tasks 255
TaskTerminated event type 440
TaskUnlocked event type 441
TaskVisibilityChange event type 442
template expressions 171
template values 168
TerminateReview event type 443
testing process versions:deleting recordings 390
testing process versions:event catches 397
testing process versions:invoking 385
testing process versions:playback mode 386
testing process versions:playing back recording 386
testing process versions:playing recordings 388
testing process versions:stopping and starting recording 384
testing process versions:test cases 394
testing process versions:using realistic scenarios 393
testing process versions:using the Stall service 398
testing process versions:Variable Logger service 396
testing, environment considerations 393
text, on process diagrams 133
time-equal XPath function 325
time-greater-than XPath function 325
time-less-than XPath function 326
time-zone identifiers 341
titles, process cards 268
toImage2 operation:Convert PDF service 899
ToImageOptionsSpec variable type 734
toPS2 operation:Convert PDF service 903
ToPSOptionsSpec variable type 737
transactional branches 185
transactions:about 187
transactions:branches 188
TransferModeEnum variable type 742
Transform operation:XSLT Transformation service 1308
Transform using XSLT(URL) operation:XSLT Transformation service 1309
TransformationFormat variable type 742
transformPDF operation:Output service 1101
TransformTo variable type 743
translate XPath function 338
TransportSecurityEnum variable type 745
true XPath function 300
TSPOptionSpec variable type 746
types of forms 216

## U

undeploying:process 138
uninstalling components 72
Unlock Certificate Encrypted PDF operation:Encryption service 943
unlock Password Encrypted PDF operation:Encryption service 944
Unlock policy protected PDF operation:Rights Management service 1142
Unrevoke license operation:Rights Management service 1142
Update Item operation:IBM Content Manager connector 852
Update policy operation:Rights Management service 1143
update_time, process property 77
UpdateVersionType variable type 749
upper-case XPath function 339
URLSpec variable type 750
User Lookup service 1280
user roles 33
User service 1255
User service:related event types 409
User service:role-based task assignment 96
UserManagerLookup Group variable type 556
UserManagerLookup User variable type 752
users:assigning tasks 232, 233

## V

validatiing process versions:customizing behavior

363
validatiing process versions:validation message reference 364
validatiing process versions:validation messages 362
validation messages:about 362
validation messages:reference 364
Variable Logger service 1288
Variable Logger service:about 1288
Variable Logger service:debugging process versions 396
variables:as property values 168
variables:common properties 505
variables:creating 163
variables:General property group 506
variables:implicit properties 505
variables:reusing 80
VerificationTime variable type 756
Verify PDF Document operation:Signature service 1211
Verify PDF Signature (deprecated) operation:Signature service 1228
Verify PDF Signature2 operation:Signature service 1219
Verify XML Signature operation:Signature service 1237
versions:files, retrieving old versions 65
versions:resource history 64
versions:resources 64
views:changing the location 72
views:Components 69
views:Form Design perspective 54
views:Process Design perspective 74
views:Resources 62

## W

Wait Point service 1290
Wait Point service:about 1290
watched folder start point 131
Watched folders, invoking processes 386
Web Service service 1291
Web Service Settings dialog box 1293
workspace start point 128, 129
Write Document operation:File Utilities service 969
Write String operation:File Utilities service 971
Write XML operation:File Utilities service 973

## X

xfaForm variable type 757
xfaForm variable type:datatype specific settings 757
xfaForm variable type:render service 758
xfaForm variable type:schema settings 760
xfaForm variable type:submit service 758
XML Document Info Editor dialog box 1056
xml variable type 760
XMLFormat variable type 761
XMLSignatureVerificationResult variable type 762
XMP Utilities service 1304
XMP Utilities service:about 1304
XMPUtilityMetadata variable type 762
XPath Builder:about 279
XPath Builder:displaying form fields 219
XPath Builder:editing text 282
XPath Builder:simple expressions 283
XPath Builder:using functions 285
XPath Builder:using operators 288
XPath expressions, as property values 168
XPath Expressions, functions 285
XPath expressions:collections 295
XPath expressions:converting data types 294
XPath expressions:form field values 291
XPath Expressions:list indexes 297
XPath expressions:map keys 297
XPath expressions:modifying form data 292
XPath expressions:node sets 293
XPath expressions:operators 288
XPath Expressions:simple 283
XPath expressions:XPath Builder 279
XPath functions:special characters 298
XSLT script:transforming the XML source 1311
XSLT Source Text Input and Testing dialog box 1311
XSLT Transformation service 1308
XSLT Transformation service:about 1308
XSLT Transformation service:configuring 1308

## Z

zooming 132