

# ADOBE® PRIMETIME TVSDK 2.0

## DIGITAL HOME PORTING KIT FOR JAVASCRIPT

---

Advertising APIs .....	4
TimedMetadata .....	4
AdSignalingMode .....	4
AdvertisingMetadata .....	5
CustomRangeMetadata / CustomRangeType .....	5
ReplaceTimeRange .....	5
Placement / PlacementType .....	6
Opportunity .....	6
Reservation .....	6
Timeline / TimelineItem / TimelineMarker .....	7
ContentTracker .....	7
AdBreak .....	7
Ad / AdAsset / AdClick / AdList / AdAssetList / AdBannerAsset .....	8
AdBreakTimelineItem / AdTimelineItem / AdBreakTimelineItemList .....	9
AdBreakPolicy / AdBreakWatchedPolicy / AdPolicy / AdPolicyMode / AdPolicyInfo / AdPolicySelector .....	9
TimelineOperation .....	11
AdBreakPlacement .....	11
AuditudeSettings .....	11
Customization .....	12
MediaPlayerItemConfig .....	12
ContentFactory .....	12
NetworkConfiguration .....	12
DRM .....	13
DRM Workflow Initialization for 2.0 .....	13
DRMAcquireLicenseSettings / DRMAuthenticationMethod .....	13
DRMMetadata .....	13
DRMPlaybackTimeWindow .....	14
DRMLicense .....	14
DRMLicenseDomain .....	14
DRMPolicy .....	14
DRMManager .....	15

Generic Playback .....	16
MediaResource.....	16
MediaPlayer .....	16
ABRControlParameters .....	19
BufferControlParameters.....	20
TextFormat .....	20
MediaPlayerItemLoader .....	22
Media Characteristics .....	22
MediaPlayerItem.....	22
Track / AudioTrack / AudioTrackList / ClosedCaptionsTrack / ClosedCaptionsTrackList .....	23
Profile / ProfileList .....	24
DRMMetadataInfo / DRMMetadataInfoList.....	24

# Advertising APIs

## TimedMetadata

class/enum	2.0 API
TimedMetadata	<pre>interface TimedMetadata {     const unsigned short METADATA_TYPE_TAG = 0 ;     const unsigned short METADATA_TYPE_ID3 = 1 ;     readonly attribute unsigned short type;      readonly attribute long time;     readonly attribute DomString id;     readonly attribute DomString name;     readonly attribute DomString content;     readonly attribute Object metadata; };</pre>
TimedMetadataList	<pre>interface TimedMetadataList {     readonly attribute unsigned long length;     getter TimedMetadata(unsigned long index); };</pre>

## AdSignalingMode

class/enum	2.0 API
AdSignalingMode	<pre>Interface AdSignalingMode {     const unsigned short MODE_DEFAULT,     const unsigned short MODE_MANIFEST_CUES,     const unsigned short MODE_SERVER_MAP,     const unsigned short MODE_CUSTOM_RANGES };</pre>

## AdvertisingMetadata

class/enum	2.0 API
AdvertisingMetadata	<pre>Interface AdvertisingMetadata {     attribute AdSignalingMode mode;     attribute AdBreakWatchedPolicy adBreakAsWatched;     attribute boolean livePreroll;     attribute boolean delayAdLoading ; };</pre>

## CustomRangeMetadata / CustomRangeType

class/enum	2.0 API
CustomRangeMetadata	<pre>Interface CustomRangeMetadata {     const unsigned short TYPE_MARK_RANGE;     const unsigned short TYPE_DELETE_RANGE;     const unsigned short TYPE_REPLACE_RANGE;     attribute unsigned short type;     attribute boolean adjustSeekPosition;     attribute TimeRangeList timeRangeList; };</pre>

## ReplaceTimeRange

class/enum	2.0 API
ReplaceTimeRange	<pre>interface ReplaceTimeRange {     attribute unsigned long begin;     readonly attribute unsigned long end;     attribute unsigned long duration;     attribute unsigned long replaceDuration; };</pre>

## Placement / PlacementType

class/enum	2.0 API
Placement	<pre>Interface Placement {     const unsigned short TYPE_MID_ROLL;     const unsigned short TYPE_PRE_ROLL;     const unsigned short TYPE_POST_ROLL;     const unsigned short TYPE_SERVER_MAP;     const unsigned short TYPE_CUSTOM_RANGE;     readonly attribute unsigned short type;     readonly attribute long time;     readonly attribute long duration;     const unsigned short MODE_DEFAULT;     const unsigned short MODE_INSERT;     const unsigned short MODE_REPLACE;     const unsigned short MODE_DELETE;     const unsigned short MODE_MARK;     const unsigned short MODE_FREE_REPLACE;     readonly attribute unsigned short mode;     readonly attribute TimeRange range; };</pre>

## Opportunity

class/enum	2.0 API
Opportunity	<pre>interface Opportunity {     readonly attribute DomString id;     readonly attribute Placement placement;     readonly attribute Object settings;     readonly attribute Object customParameters; };</pre>

## Reservation

class/enum	2.0 API
Reservation	<pre>interface Reservation {     readonly attribute TimeRange range;     readonly attribute long hold; };</pre>

## Timeline / TimelineItem / TimelineMarker

class/enum	2.0 API
Timeline	<pre>interface Timeline {     readonly attribute TimelineMarkerList timelineMarkers;     readonly attribute TimelineItemList timelineItems;      double convertToLocalTime(double time);     double convertToVirtualTime(double time); };</pre>
TimelineItem	<pre>interface TimelineItem : TimelineMarker {     readonly attribute long id;     readonly attribute TimeRange virtualRange;     readonly attribute TimeRange localRange;     readonly attribute boolean watched;     readonly attribute boolean temporary; };</pre>
TimelineMarker	<pre>interface TimelineMarker {     readonly attribute double time;     readonly attribute double duration; };</pre>

## ContentTracker

class/enum	2.0 API
ContentTracker	NA

## AdBreak

class/enum	2.0 API
AdBreak	<pre>interface AdBreak {     readonly attribute double duration;     readonly attribute AdList ads;     readonly attribute AdInsertionType insertionType; };</pre>

## Ad / AdAsset / AdClick / AdList / AdAssetList / AdBannerAsset

class/enum	2.0 API
Ad	<pre> interface Ad {     readonly attribute AdAsset primaryAsset;     readonly attribute AdAssetList companionAssets;     readonly attribute double duration;     readonly attribute DomString id;     const unsigned short ADTYPE_LINEAR = 0 ;     const unsigned short ADTYPE_NONLINEAR = 1 ;     readonly attribute unsigned short adType;     readonly attribute AdInsertionType adInsertionType;     readonly attribute boolean clickable;     readonly attribute boolean isCustomAdMarker; }; </pre>
AdAsset	<pre> interface AdAsset {     readonly attribute DomString id;     readonly attribute double duration;     readonly attribute MediaResource resource;     readonly attribute AdClick adClick;     readonly attribute Object metadata; }; </pre>
AdClick	<pre> interface AdClick {     readonly attribute DomString id;     readonly attribute DomString title;     readonly attribute DomString url; }; </pre>
AdList	<pre> interface AdList    {     readonly attribute unsigned long length;     getter Ad(unsigned long index); }; </pre>
AdAssetList	<pre> interface AdAssetList {     readonly attribute unsigned long length;     getter AdAsset(unsigned long index); }; </pre>
AdBannerAsset	<pre> interface AdBannerAsset : AdAsset {     readonly attribute int width;     readonly attribute int height;     readonly attribute DomString staticUrl;     readonly attribute DomString height;     readonly attribute DomString width; }; </pre>

## AdBreakTimelineItem / AdTimelineItem / AdBreakTimelineItemList

class/enum	2.0 API
AdBreakTimelineItem	<pre>interface AdBreakTimelineItem : TimelineItem {     readonly attribute AdBreak adBreak;     readonly attribute AdTimelineItemList items; };</pre>
AdTimelineItem	<pre>interface AdTimelineItem : TimelineItem {     readonly attribute AdBreak adBreak;     readonly attribute Ad ad; };</pre>
AdBreakTimelineItemList	<pre>interface AdBreakTimelineItemList {     readonly attribute unsigned long length;     getter AdBreakTimelineItem (unsigned long index); };</pre>

## AdBreakPolicy / AdBreakWatchedPolicy / AdPolicy / AdPolicyMode / AdPolicyInfo / AdPolicySelector

class/enum	2.0 API
AdBreakPolicy	<pre>interface AdBreakPolicy {     readonly attribute short AD_BREAK_POLICY_SKIP;     readonly attribute short AD_BREAK_POLICY_PLAY;     readonly attribute short AD_BREAK_POLICY_REMOVE;     readonly attribute short AD_BREAK_POLICY_REMOVE_AFTER_PLAY; };</pre>
AdBreakWatchedPolicy	<pre>interface AdBreakWatchedPolicy {     readonly attribute short AD_BREAK_AS_WATCHED_ON_BEGIN;     readonly attribute short AD_BREAK_AS_WATCHED_ON_END;     readonly attribute short AD_BREAK_AS_WATCHED_NEVER; };</pre>
AdPolicy	<pre>interface AdPolicy {     readonly attribute short AD_POLICY_PLAY;     readonly attribute short AD_POLICY_PLAY_FROM_AD_BEGIN;     readonly attribute short AD_POLICY_PLAY_FROM_AD_BREAK_BEGIN;     readonly attribute short AD_POLICY_SKIP_TO_NEXT_AD_IN_BREAK;     readonly attribute short AD_POLICY_SKIP_AD_BREAK; };</pre>

**AdPolicyMode**

```
interface AdPolicyMode {  
    readonly attribute short AD_POLICY_MODE_PLAY;  
    readonly attribute short AD_POLICY_MODE_SEEK;  
    readonly attribute short AD_POLICY_MODE_TRICKPLAY;  
};
```

**AdPolicyInfo**

```
interface AdPolicyInfo {  
    readonly attribute AdBreakTimelineItemList  
        adBreakTimelineItems;  
    readonly attribute AdTimelineItem adTimelineItem;  
    readonly attribute double currentTime;  
    readonly attribute double seekToTime;  
    readonly attribute double rate;  
    readonly attribute short mode; //AdPolicyMode  
};
```

**AdPolicySelector**

```
interface AdPolicySelector {  
    /**  
     * AdbreakPolicy selectPolicyForAdBreak(  
     *     AdPolicyInfo adPolicyInfo);  
     */  
    attribute Object selectPolicyForAdBreakCallbackFunc;  
  
    /**  
     * AdBreakTimelineItemList selectAdBreaksToPlay(  
     *     AdPolicyInfo adPolicyInfo);  
     */  
    attribute Object selectAdBreaksToPlayCallbackFunc;  
    /**  
     * AdPolicy selectPolicyForSeekIntoAd(  
     *     AdPolicyInfo adPolicyInfo);  
     */  
    attribute Object selectPolicyForSeekIntoAdCallbackFunc;  
    /**  
     * AdBreakWatchedPolicyselectWatchedPolicyForAdBreak(  
     *     AdPolicyInfo adPolicyInfo);  
     */  
    attribute Object selectWatchedPolicyForAdBreakCallbackFunc;  
};
```

## TimelineOperation

class/enum	2.0 API
TimelineOperation	<pre>interface TimelineOperation {     readonly attribute Placement placement ; };</pre>

## AdBreakPlacement

class/enum	2.0 API
AdBreakPlacement	<pre>interface AdBreakPlacement : TimelineOperation {     readonly attribute AdBreak adBreak;     readonly attribute Placement placement; // From TimelineOperation     readonly attribute double time;     readonly attribute double duration; };</pre>

## AuditdeSettings

class/enum	2.0 API
AuditdeSettings	<pre>interface AuditdeSettings : AdvertisingMetadata{     attribute DomString zoneId;     attribute DomString mediaId;     attribute DomString defaultMediaId ;     attribute DomString domain ;     attribute Object targettingInfo ;     attribute Object customParameters ;     attribute Boolean creativePackaingEnabled ;     attribute Boolean showStaticBanners; };</pre>

# Customization

## MediaPlayerItemConfig

class/enum	2.0 API
MediaPlayerItemConfig	<pre>interface MediaPlayerItemConfig {     attribute ContentFactory adFactory;     attribute StringList subscribeTags;     attribute StringList adTags;     attribute AdSignalingMode adSignalingMode;     attribute CustomRangeMetadata customRangeMetadata;     attribute NetworkConfiguration networkConfiguration;     attribute AdvertisingMetadata advertisingMetadata;     attribute Boolean useHardwareDecoder; };</pre>

## ContentFactory

class/enum	2.0 API
ContentFactory	<pre>interface ContentFactory {     /*      * AdPolicySelector retrieveAdPolicySelector(MediaPlayerItem item);      */     attribute Object retrieveAdPolicySelectorCallbackFunc; };</pre>

## NetworkConfiguration

class/enum	2.0 API
NetworkConfiguration	<pre>interface NetworkConfiguration {     attribute boolean forceNativeNetworking;     attribute boolean useRedirectedUrl;     attribute Object cookieHeader;     attribute boolean readSetCookieHeader;     attribute int masterUpdateInterval;     attribute boolean useCookieHeaderForAllRequests;     attribute int readLimit; };</pre>

# DRM

## DRM Workflow Initialization for 2.0

JS application needs to call `AdobePSDK.initiateDRMWorkflow`. Otherwise, DRM videos will not play.

class/enum	2.0 API
AdobePSDK	<pre>interface AdobePSDK {     void initiateDRMWorkflow(         DomString appStoratePath,         DomString publisherId,         DomString appId,         DomString appVersion,         boolean privacyModeOn); };</pre>

## DRMAcquireLicenseSettings / DRMAuthenticationMethod

class/enum	2.0 API
DRMAcquireLicenseSettings	<pre>enum DRMAcquireLicenseSettings {     const unsigned int FORCE_REFRESH = 0;     const unsigned int LOCAL_ONLY = 1;     const unsigned int ALLOW_SERVER = 2; };</pre>
DRMAuthenticationMethod	<pre>enum DRMAuthenticationMethod {     const unsigned int UNKNOWN = 0;     const unsigned int ANONYMOUS = 1;     const unsigned int USERNAME_AND_PASSWORD = 2; };</pre>

## DRMMetadata

class/enum	2.0 API
DRMMetadata	<pre>interface DRMMetadata {     readonly attribute DomString serverUrl;     readonly attribute DomString licenseId;     readonly attribute DRMPolicyArray policies; };</pre>

## DRMPlaybackTimeWindow

class/enum	2.0 API
------------	---------

DRMPlaybackTimeWindow	<pre>interface DRMPlaybackTimeWindow {     readonly attribute int playbackPeriodInSeconds;     readonly attribute long playbackStartDate;     readonly attribute long playbackEndDate; };</pre>
-----------------------	---

## DRMLicense

class/enum	2.0 API
------------	---------

DRMLicense	<pre>interface DRMLicense {     readonly attribute Uint8Array bytes;     readonly attribute Date licenseStartDate;     readonly attribute Date licenseEndDate;     readonly attribute Date offlineStorageStartDate;     readonly attribute Date offlineStorageEndDate;     readonly attribute DomString serverUrl;     readonly attribute DomString licenseID;     readonly attribute DomString policyID;     readonly attribute DRMPlaybackTimeWindow playbackTimeWindow;     readonly attribute Object customProperties; };</pre>
------------	---

## DRMLicenseDomain

class/enum	2.0 APIs
------------	----------

DRMLicenseDomain	<pre>interface DRMLicenseDomain {     readonly attribute DomString authenticationDomain;     readonly attribute DRMAuthenticationMethod authenticationMethod;     readonly attribute DomString serverUrl; };</pre>
------------------	--

## DRMPolicy

class/enum	2.0 API
------------	---------

DRMPolicy	<pre>interface DRMPolicy {     readonly attribute DomString authenticationDomain;     readonly attribute DRMAuthenticationMethod authenticationMethod;     readonly attribute DomString displayName;     readonly attribute DRMLicenseDomain licenseDomain; };</pre>
-----------	--

## DRMManager

### class/enum

### 2.0 API

#### DRMManager

```
interface DRMManager ← EventTarget
{
    void acquireLicense(DRMMetadata metadata,
        DRMAcquireLicenseSettings setting,
        DRMAquireLicenseListener listener);
    void acquirePreviewLicense(DRMMetadata metadata,
        DRMAquireLicenseListener listener);
    void authenticate(DRMMetadata metadata,
        DomString url, DomString &authenticationDomain,
        DomString user, DomString password,
        DRMAuthenticateListener listener);
    DRMMetadata createMetadataFromBytes(UINT8Array array,
        DRMErrorListener listener);
    void initialize(DRMOperationCompleteListener listener);
    attribute long maxOperationTime;
    void joinLicenseDomain(DRMLicenseDomain licenseDomain,
        boolean forceRefresh,
        DRMOperationCompleteListener listener);
    void leaveLicenseDomain(DRMLicenseDomain licenseDomain,
        DRMOperationCompleteListener listener);
    void resetDRM(DRMOperationCompleteListener listener);
    void returnLicense(DomString serverURL,
        DomString licenseID, DomString policyID,
        boolean commitImmediately,
        DRMReturnLicenseListener listener);
    void setAuthenticationToken(DRMMetadata metadata,
        DomString authenticationDomain, UINT8Array token,
        DRMOperationCompleteListener listener);
    void storeLicenseBytes(UINT8Array licenseBytes,
        DRMOperationCompleteListener listener);
};
```

### Events supported by DRMManager

#### DRMErrorListener

```
interface DRMErrorListener {
    /*
     * void onDRMErrorCallbackFunc(uint major, uint minor,
     *                               DomString errorString, DomString errorServerUrl) ;
     */
    attribute Object onDRMErrorCallbackFunc;
}
```

#### DRMOperationCompleteListener

```
interface DRMOperationCompleteListener : DRMErrorListener {
    // void onDRMOperationComplete();
    attribute Object onDRMOperationCompleteCallbackFunc;
};
```

#### DRMAuthenticateListener

```
interface DRMAuthenticateListener : DRMErrorListener {
    // void onAuthenticationComplete(
    //     UINT8Array authenticationToken);
    attribute Object onAuthenticationCompleteCallbackFunc;
}
```

DRMAquireLicenseListener	interface DRMAquireLicenseListener : DRMErrorListener { // void onLicenseAcquired(DRMLicense); attribute Object onLicenseAcquiredCallbackFunc; };
DRMReturnLicenseListener	interface DRMReturnLicenseListener : DRMErrorListener { // void onLicenseReturnComplete(uint numReturned ); attribute Object onLicenseReturnCompleteCallbackFunc; };

## Generic Playback

### MediaResource

Class/enum	2.0 API
MediaResource	interface MediaResource { attribute DomString url; attribute unsigned short type; attribute Object metadata; const unsigned short TYPE_HLS; const unsigned short TYPE_HDS; const unsigned short TYPE_DASH; const unsigned short TYPE_CUSTOM; const unsigned short TYPE_UNKNOWN; };

### MediaPlayer

class/enum	2.0 API
MediaPlayer	interface MediaPlayer : EventTarget{ void prepareToPlay( double position); void play(); void pause(); void seek( double position); void seekToLocal( double position); void reset(); void release(); void replaceCurrentItem(MediaPlayerItem item); void replaceCurrentResource(MediaResource rsource, MediaPlayerItemConfig config); void suspend(); void restore(); void notifyClick(); readonly attribute TimeRange playbackRange;

```

readonly attribute TimeRange seekableRange;
readonly attribute double currentTime;
readonly attribute double localTime;
readonly attribute TimeRange bufferedRange;
readonly attribute DRMManager drmManager;
readonly attribute MediaPlayerItem currentItem;

readonly attribute unsigned short status;

attribute unsigned short volume;
attribute ABRControlParameters abrControlParameters;
attribute BufferControlParameters bufferControlParameters;
const unsigned short VISIBLE; //For CC visibility
const unsigned short INVISIBLE; //For CC visibility
attribute unsigned short ccVisibility;
attribute TextFormat ccStyle;
readonly attribute PlaybackMetrics playbackMetrics;
        attribute double rate;
        attribute MediaPlayerView view;
readonly attribute Timeline timeline;
        attribute double currentTimeUpdateInterval; // setting this
Won't be supported for 2.0
};

interface MediaPlayerStatus {
// PlayerStatus
    const unsigned short PLAYER_STATUS_IDLE;
    const unsigned short PLAYER_STATUS_INITIALIZING;
    const unsigned short PLAYER_STATUS_INITIALIZED;
    const unsigned short PLAYER_STATUS_PREPARING;
    const unsigned short PLAYER_STATUS_PREPARED;
    const unsigned short PLAYER_STATUS_PLAYING;
    const unsigned short PLAYER_STATUS_PAUSED;
    const unsigned short PLAYER_STATUS SEEKING;
    const unsigned short PLAYER_STATUS_COMPLETE;
    const unsigned short PLAYER_STATUS_ERROR;
    const unsigned short PLAYER_STATUS_RELEASED;
    const unsigned short PLAYER_STATUS_SUSPENDED;
};

```

Events  
supported by  
MediaPlayer

Event Name	Interface	Fired when
itemUpdated	Event	When the media player has

class/enum	2.0 API		
			successfully updated the media.
	timedMetadataAvailable	TimedMetadataEvent	When a new timed metadata is discovered in the manifest.
	timelineUpdated	Event	When the media player has an updated timeline.
	statusChanged	StatusEvent	When the state of media player has changed.
	sizeAvailable	SizeEvent	When the size of media is available
	adBreakStarted	AdBreakEvent	When an ad break has starts.
	adStarted	AdEvent	When an ad starts
	adProgress	AdProgressEvent	When an ad is in progress. Fired to report % ad progress.
	timeChanged	TimeEvent	When media playback is in progress, Fired to report the % playback progress.
	adCompleted	AdEvent	When an ad has been played completely.
	adBreakCompleted	AdBreakEvent	When an ad break has been played completely.
	bufferingBegin	Event	When buffering has started.
	bufferingEnd	Event	When buffering is complete.
	seekBegin	SeekEvent	When seek is starting.
	seekEnd	SeekEvent	When seeking is complete.
	loadInformationAvailable	LoadInformationEvent	When a fragment that was successfully downloaded.
	operationFailed	NotificationEvent	When a recoverable error occurs.
	drmMetadataInfoAvailable	DRMMetadataEvent	When a new drm metadata is available.
	reservationReached	ReservationrEvent	When buffering reaches a timeline area affected by the reservation and the buffering is paused.
	rateSelected	PlaybackRateEvent	When the trickPlay operation is initiated.
	ratePlaying	PlaybackRateEvent	When the content is actually playing in the selected rate.
	adBreakSkipped	AdBreakEvent	When an ad break is skipped due to

class/enum	2.0 API		
			trick play.
	adClicked	AdClickedEvent	When user clicks on an Ad.
	profileChanged	ProfileEvent	When the playback profile is changed.
	seekPositionAdjusted	SeekEvent	When seek position is adjusted due internal or external rules.
	audioUpdated	MediaPlayerItemEvent	When a media player item is updated. For certain streams that contain audio tracks that are only detectable at playback time, this event is fired when new audio tracks are available.
	captionUpdated	MediaPlayerItemEvent	When a media player item is updated. For live/linear streams the client must periodically refresh the media resource to detect the new available content. When this happen certain media characteristics might change.
	masterUpdated	MediaPlayerItemEvent	When a media player item is updated. For live/linear streams the client must periodically refresh the media resource to detect the new available content. When this happen certain media characteristics might change.
	playbackRangeUpdated	MediaPlayerItemEvent	When a media player item is updated. For live/linear streams the client must periodically refresh the media resource to detect the new available content. When this happen certain media characteristics might change.
	timedEvent	TimedEvent	Sent when timed events are generated  See spec: PSDK/AVE Timed Events

## ABRControlParameters

class/enum	2.0 API
ABRControlParameters	<pre> interface ABRControlParameters {     const unsigned short ABR_POLICY_CONSERVATIVE = 0 ;     const unsigned short ABR_POLICY_MODERATE = 1 ;     const unsigned short ABR_POLICY_AGGRESIVE = 2 ;      attribute unsigned short abrPolicy;     attribute unsigned int initialBitRate; </pre>

**class/enum****2.0 API**

```
    attribute unsigned int minBitRate;
    attribute unsigned int maxBitRate;
    const unsigned short DEFAULT_ABR_INITIAL_BITRATE;
    const unsigned short DEFAULT_ABR_MIN_BITRATE;
    const unsigned short DEFAULT_ABR_MAX_BITRATE;
    const ABRPolicy DEFAULT_ABR_POLICY;
};
```

## BufferControlParameters

**class/enum****2.0 API****BufferControlParameters**

```
interface BufferControlParameters {
    attribute double initialBufferTime;
    attribute double playBufferTime;
    const double DEFAULT_INITIAL_BUFFER_TIME;
    const double DEFAULT_PLAY_BUFFER_TIME;
};
```

## TextFormat

**class/enum****2.0 API****TextFormat**

```
interface TextFormat{
    // Color
    const unsigned short COLOR_DEFAULT = 0 ;
    const unsigned short COLOR_BLACK = 1 ;
    const unsigned short COLOR_GRAY = 2 ;
    const unsigned short COLOR_WHITE = 3 ;
    const unsigned short COLOR_BRIGHT_WHITE = 4 ;
    const unsigned short COLOR_DARK_RED = 5 ;
    const unsigned short COLOR_RED = 6 ;
    const unsigned short COLOR_BRIGHT_RED = 7 ;
    const unsigned short COLOR_DARK_GREEN = 8 ;
    const unsigned short COLOR_GREEN = 9 ;
    const unsigned short COLOR_BRIGHT_GREEN = 10 ;
    const unsigned short COLOR_DARK_BLUE = 11 ;
    const unsigned short COLOR_BLUE = 12 ;
    const unsigned short COLOR_BRIGHT_BLUE = 13 ;
    const unsigned short COLOR_DARK_YELLOW = 14 ;
    const unsigned short COLOR_YELLOW = 15 ;
    const unsigned short COLOR_BRIGHT_YELLOW = 16 ;
};
```

```
const unsigned short COLOR_DARK_MAGENTA = 17 ;
const unsigned short COLOR_MAGENTA = 18 ;
const unsigned short COLOR_BRIGHT_MAGENTA = 19 ;
const unsigned short COLOR_DARK_CYAN = 20 ;
const unsigned short COLOR_CYAN = 21 ;
const unsigned short COLOR_BRIGHT_CYAN = 22 ;

readonly attribute unsigned short fontColor;
readonly attribute unsigned short backgroundColor;
readonly attribute unsigned short fillColor;
readonly attribute unsigned short edgeColor;

// Size
const unsigned short SIZE_DEFAULT = 0 ;
const unsigned short SIZE_SMALL = 1 ;
const unsigned short SIZE_MEDIUM = 2 ;
const unsigned short SIZE_LARGE = 3 ;

readonly attribute unsigned short size;

// FontEdge
const unsigned short FONT_EDGE_DEFAULT = 0 ;
const unsigned short FONT_EDGE_NONE = 1 ;
const unsigned short FONT_EDGE_RAISED = 2 ;
const unsigned short FONT_EDGE_DEPRESSED = 3 ;
const unsigned short FONT_EDGE_UNIFORM = 4 ;
const unsigned short FONT_EDGE_DROP_SHADOW_LEFT = 5 ;
const unsigned short FONT_EDGE_DROP_SHADOW_RIGHT = 6 ;
readonly attribute unsigned short fontEdge;

// Font
const unsigned short FONT_DEFAULT = 0 ;
const unsigned short FONT_MONOSPACED_WITH_SERIFS = 1 ;
const unsigned short FONT_PROPORTIONAL_WITH_SERIFS = 2 ;
const unsigned short FONT_MONOSPACED_WITHOUT_SERIFS = 3 ;
const unsigned short FONT_CASUAL = 4 ;
const unsigned short FONT_CURSIVE = 5 ;
const unsigned short FONT_SMALL_CAPITALS = 6 ;
readonly attribute unsigned short font;
readonly attribute unsigned short fontOpacity;
readonly attribute unsigned short backgroundOpacity;
readonly attribute unsigned short fillOpacity;
readonly attribute unsigned short DEFAULT_OPACITY;

};
```

## MediaPlayerItemLoader

class/enum	2.0 API
MediaPlayerItemLoader	<pre>interface MediaPlayerItemLoader: {     void load(MediaResource resource, long resourceId,               ItemLoaderListener listener,               MediaPlayerItemConfig config) ;     void cancel();     readonly attribute MediaPlayerItem currentItem; };</pre>
ItemLoaderListener	<pre>interface ItemLoaderListener { //onLoadCompleteCallbackFunc(MediaPlayerItem)     var onLoadCompleteCallbackFunc; //onErrorCallbackFunc(PSDKErrorCode)     var onErrorCallbackFunc; }</pre>

## Media Characteristics

### MediaPlayerItem

class/enum	2.0 API
MediaPlayerItem	<pre>interface MediaPlayerItem {     readonly attribute MediaResource resource;     readonly attribute long resourceId;     readonly attribute boolean live;     readonly attribute boolean hasAlternateAudio;     readonly attribute AudioTrackList audioTracks;     readonly attribute AudioTrack selectedAudioTrack;     void selectAudioTrack(AudioTrack track);     readonly attribute boolean hasClosedCaptions;     readonly attribute ClosedCaptionsTrackList closedCaptionsTracks;     readonly attribute ClosedCaptionsTrack selectedClosedCaptionsTrack;     void selectClosedCaptionsTrack(ClosedCaptionsTrack track);      readonly attribute boolean hasTimedMetadata;     readonly attribute TimedMetadataList timedMetadata;     readonly attribute boolean dynamic;     readonly attribute boolean isProtected;     readonly attribute DRMMetadataInfoList drmMetadataInfos;     readonly attribute ProfileList profiles;</pre>

class/enum	2.0 API
	<pre> readonly attribute Profile selectedProfile; readonly attribute boolean trickPlaySupported; readonly attribute FloatArray availablePlaybackRates; readonly attribute float selectedPlaybackRate; readonly attribute MediaPlayer mediaPlayer; readonly attribute MediaPlayerIyemConfig config; }; </pre>

## Track / AudioTrack / AudioTrackList / ClosedCaptionsTrack / ClosedCaptionsTrackList

class/enum	2.0 API
Track	<pre> interface Track{     readonly attribute DomString name;     readonly attribute DomString language;     readonly attribute boolean default;     readonly attribute boolean autoSelect; }; </pre>
AudioTrack	<pre> interface AudioTrack : Track {     readonly attribute DomString name; //FromTrack     readonly attribute DomString language;//FromTrack     readonly attribute boolean default; // From Track     readonly attribute boolean autoSelect;//FromTrack     readonly attribute unsigned int pid; }; </pre>
AudioTrackList	<pre> interface AudioTrackList {     readonly attribute unsigned long length;     getter AudioTrack (unsigned long index); }; </pre>
ClosedCaptionsTrack	<pre> interface ClosedCaptionsTrack : Track {     readonly attribute DomString name; //FromTrack     readonly attribute DomString language;//FromTrack     readonly attribute boolean default; // FromTrack     readonly attribute boolean autoSelect;//FromTrack      const unsigned short SERVICE_608_CAPTIONS = 0;     const unsigned short SERVICE_708_CAPTIONS = 1;     const unsigned short SERVICE_WEB_VTT_CAPTIONS = 2;     readonly attribute unsigned short serviceType;     readonly attribute boolean forced; }; </pre>

class/enum	2.0 API
	<code>};</code>
ClosedCaptionsTrackList	<pre>interface ClosedCaptionsTrackList {     readonly attribute unsigned long length;     getter ClosedCaptionsTrack(unsigned long index); };</pre>

## Profile / ProfileList

class/enum	2.0 API
Profile	<pre>interface Profile {     readonly attribute unsigned int width;     readonly attribute unsigned int height;     readonly attribute unsigned int bitRate; };</pre>
ProfileList	<pre>interface ProfileList {     readonly attribute unsigned long length;     getter Profile(unsigned long index); };</pre>

## DRMMetadataInfo / DRMMetadataInfoList

class/enum	2.0 API
DRMMetadataInfo	<pre>interface DRMMetadataInfo {     readonly attribute DRMMetadata metadata;     readonly attribute long prefetchTimestamp;     readonly attribute TimeRange timeRange; };</pre>
DRMMetadataInfoList	<pre>interface DRMMetadataInfoList {     readonly attribute unsigned long length;     getter DRMMetadataInfo(unsigned long index); };</pre>