Widgets for Adobe® Acrobat® : VideoPlayerX

The Enhanced Video Tool for Adobe Acrobat & InDesign

Current build is VP10.40, issued 28 Oct 2014

The Video Tool in Adobe Acrobat 9+ and InDesign CS5+ allows a Flash video (FLV or H.264) to be embedded into a PDF document. It achieves this by embedding both the video file and a SWF widget, called VideoPlayer.swf, which is part of your Acrobat or InDesign installation package. The default widget has bare-bones functionality and does not support things like looping, event triggers or track changes.

In response to user requests, we are enhancing the widget with new features, and from this page you can access the current builds and documentation. We rely on your input to steer the development process, so please send us your comments and suggestions - they will be incorporated into the builds subject to approval. Should you encounter an error, please report it to us.

When the VideoPlayerX widget is used, the modified SWF is embedded into the PDF file. It will therefore play when opened on any standard install of Adobe Acrobat or Adobe Reader 9.2 and above - your viewers do not need to download the widget itself.

New features :: DOWNLOAD :: New API calls :: Release notes :: Licensing

DOWNLOAD

New Features

VideoPlayerX adds a number of new features to both the visual playback system and the JavaScript API:

Change video source via Acrobat JavaScript
VideoPlayerX adds several new API calls (as below) including the ability to specify a new video source (either local or remote), allowing more than one video to be played in the same annotation.

Dynamic skin switching and support for "under" styles
API calls permit switching of skins, and the widget supports skin styles that sit below the video area rather than overlaying it. To add additional skins, place them in the Multimedia Skins folder. FLVPlayback skins are supplied with Adobe Flash Professional, and we have included one example skin in the download package for you to work with.

Compensation for skin position
When a skin is used in "under" mode, the player resizes within the annotation rectangle to allow room for the skin to be shown. When an "over" skin is used, the video is resized proportionally to the annotation rectangle. The mode is determined by looking for the case-insensitive substring "over" in the skin filename.

Dynamic fallback from remote streams
When a remote source is defined using the API method multimedia_setSource() and that source cannot be played, the local source is played instead. This allows a PDF author to embed a low-quality video and offer an online high-quality alternative, but allow for situations where that online file is not available or where security permissons prevent access to the Web.

Looping controls
By default, VideoPlayerX loops the video continually (*unlike* the standard widget). A new button toggles this behavior, visible on the top left corner of the video on mouse-over. This button adopts the color and opacity of the

Installation

For Adobe Acrobat 9+

1. Navigate to your Acrobat installation folder and find the Players subfolder. On Windows this will be at:

   /Program Files/Adobe/Acrobat V.0/Acrobat/Multimedia Skins/Players

   or on a 64-bit machine, at:

   /Program Files(x86)/Adobe/Acrobat V.0/Acrobat/Multimedia Skins/Players

   where V is your base version number (9,10,11,12,etc.).

   On Mac systems, it is within "Adobe Acrobat Pro.app" in your Applications folder. Show the package contents and navigate to:

   Contents/Resources/Multimedia Skins/Players

2. Make a copy of the file "VideoPlayer.swf" in this folder - this is the original version, so don't lose it! Renaming it will suffice.
3. Open the ZIP file from the download link above, and extract the new copy of VideoPlayer.swf into the above folder.
4. Optionally, copy the "MinimaUnderPlayBackSeekCounterVolMuteNoFull.swf" file to the parent Multimedia Skins folder.
5. Start Acrobat and place your videos using the Video Tool as usual.

To uninstall, simply delete the VideoPlayer.swf file and replace it with the original copy.

For Adobe InDesign CS5+

1. Installation is the same as above, but this time navigate to your InDesign installation folder and find the multimedia subfolder. For OS X users show the package contents and browse through the Contents/Resources folder as before. On Windows for CS versions the widget will be at:

   /Program Files(x86)/Adobe/Adobe InDesign CSx/Presets/multimedia

skin, and the initial mode can be controlled via the enhanced API.

Absolute scaling

By default, videos will play at absolute 100% pixel scale if there is enough room to do so, and will revert to a scale-to-fit mode if the annotation area isn't large enough. This much-requested feature ensures that the video is not resampled, so will render in perfect quality. This scale mode can be controlled via the enhanced API.

Default doc-level scripts

If your PDF file contains a document-level JavaScript function "vpx_listener()", VideoPlayerX will send it event messages.
Click to learn how to use this feature.

If your PDF file contains a document-level JavaScript function "vpx_init()", VideoPlayerX will fire it when the annotation loads.
Click to learn how to use this feature.

On Windows for the 64-bit CC version this will be at:

/Program Files/Adobe/Adobe InDesign CC (64 bit)/Presets/multimedia
Optionally, copy the supplied skin file to the *FLVPlayback Skins* subfolder.

Licensing and Copyright

New and modified API calls

The following calls can be made using the callAS method of a rich media annotation that embeds the VideoPlayerX widget, for example:

```
getAnnotsRichMedia(pageNum)[0].callAS('multimedia_volume', 0.5);
```

Some are new, and some have new return values. All the standard calls pre-defined in Acrobat 9+ are available, here we list only those which are different in VideoPlayerX:

multimedia_seekCuePoint(cuePointName:String) : String
Seeks to the named navigation cue point in an FLV video.

Returns on success: Empty string
Returns on error: String *ERROR: xxxx* where xxx is one of the standard numeric error codes defined in ActionScript 3.0.

multimedia_pause() : Number
Pauses playback.

Returns on success: Playhead time in seconds

multimedia_volume(vol:Number) : Number
Sets volume from 0.0 to 1.0.

Returns on success: Previous volume setting.

multimedia_mute() : Number
Sets volume to 0.0.

Returns on success: Previous volume setting.

multimedia_registerCueHandler(handlername:String) : String
Registers a document-level function name to receive API calls when video cue points are encountered. Set to an empty string to clear the handler. Default actions are not overridden.

The handler function will be called with five parameters (RMA,Page,Name,Type,Time):

```
RMA     name of the Rich Media Annotation
Page    PDF page on which the RMA is located
Name    cue point name
Type    "navigation" or "event"
Time    cue point time in seconds
```

Returns on success: handlername

multimedia_registerEventHandler(handlername:String) : String
Registers a document-level function name to receive API calls when video status events are triggered. Set to an empty string to clear the handler. Default actions are not overridden.

The handler function will be called with four parameters (RMA,Page,Pos,Event):

```
RMA     name of the Rich Media Annotation (string)
Page    PDF page on which the RMA is located (integer)
Pos     the playback position in seconds (float)
Event   the name of the event - such as "playing", "complete", etc.
```

Returns on success: handlername

Note that VideoPlayerX will *also* send the same parameters to the document-level function "vpx_listener()" if this function exists.

multimedia_setStageColor(color:uint) : void
Sets the background color for the Stage (the area around the video when it isn't scaled to fit the annotation).

Example: getAnnotsRichMedia(pageNum)[0].callAS("multimedia_setStageColor",0xFF00FF);

If you set your annotation to use a transparent background, this function will override it. The default color is black.

multimedia_setSkin(skinName:String) : void
Sets the player skin to that specifed.

Skin must be a local file attached to the resource dictionary of the Rich Media annotation (no prefix). This requires the annotation to be created using the Flash Tool in Acrobat, as currently the Video Tool dialog window does not allow access to the RMA resource list.

When a skin is changed from "under" to "over" mode or back using this function call, the player compensates for the change in height to ensure the skin remains visible.

multimedia_setSkinColor(color:uint) : uint
Sets the background color for the player skin (will only take effect where the skin supports color changes).

Example: getAnnotsRichMedia(pageNum)[0].callAS("multimedia_setSkinColor",0xFF0000);

Returns on success: Previous color value.

multimedia_setSkinAlpha(alpha:uint) : uint
Sets the background alpha for the player skin (will only take effect where the skin supports alpha changes).

Example: getAnnotsRichMedia(pageNum)[0].callAS("multimedia_setSkinAlpha",0.25);

Returns on success: Previous alpha value.

multimedia_getSource() : String
Returns the source of the playing video (a URL or a local file reference).

Returns on success: Source filename/URL in string format (local files have no prefix).

multimedia_setSource(url:String) : String
Sets the source for the video (a URL or a local file reference).

A remote source is identified by an http:// or Flash Media Server (rtmp, rmpte, etc) prefix. A local source is identified by a pdf:// prefix, or no prefix.

The source must be either a video file (FLV, F4V or H.264) or a SMIL resource description linking to video file(s).

Returns on success: *local=* or *remote=* and the source in string format.

If the remote source cannot be played for any reason, the player automatically returns to playing the local source instead.

multimedia_useLocal(isLocal:boolean) : String
Switches to the local source if isLocal = true, or to the remote source if isLocal = false.

Returns on success: source filename/URL in string format Returns on error: "*NOT AVAILABLE*"

multimedia_getMetdata(attribute:String) : String
Returns the video metadata associated with the attribute. Valid attribute strings are:

```
width
height
audiocodecid
videocodecid
framerate
videodatarate
duration
```

multimedia_getVideoState() : String
Returns the video state. The possible values for the state property are: "buffering", "connectionError", "disconnected", "loading", "paused", "playing", "rewinding", "seeking", and "stopped".

multimedia_setScaleMode(attribute:String) : String
Sets video scale mode. Valid attribute strings are:

```
exactFit
noScale
maintainAspectRatio
```

Returns on success: Previous value.
Note that if the scale mode is changed to "maintainAspectRatio", the align mode will be switched to "top left" rather that "center".

multimedia_isLooping() : Boolean

Sets if the video should loop automatically when it reaches the end of the timeline. Default = true

Returns on success: Previous value of the setting.

multimedia_showLoopButton(show:Boolean) : Boolean
Sets if the video loop control button should appear on mouseover (show = true) or be hidden completely (show = false). Default = true.

Note that when there is no skin, the loop button is automatically hidden.

multimedia_getVersion() : String
Returns a string in the form "*NNNN fp=FFFF vp=VVVV*"

Where NNNN is the name of the Rich Media Annotation, FFFF is the version of Flash Player being used, and VVVV is the version of the videoPlayerX code (currently 10.40). The length of each element is variable.

Note that we also accept the enhanced API calls used by Joel Geraci's VideoPlayerPlus widget, which is no longer available. VideoPlayerX is a drop-in replacement for VideoPlayerPlus.

Release notes VP10.40

1. Added the multimedia_showLoopButton() API call (also settable via FlashVars using param 'showLoopButton').

General Notes

1. The "Snap to content proportions" checkbox in Adobe Acrobat 9 takes account only of the video, and not the player skin. When using VideoPlayerX and an "under" style skin, VideoPlayerX will automatically accommodate the skin within the annotation rectangle, but you may wish to tweak the height using the Object Select tool to ensure the video fills the annotation completely.
2. It is not possible to trigger the Acrobat "full screen multimedia" viewing mode from within a SWF, therefore skins with a full screen icon should not be used.
3. The remote URL used in multimedia_setSource() can be an FLV, F4V or H.264 video hosted on an http server, a link to an Adobe Media Server stream using rtmp, rtmpe, etc., or a SMIL file referencing either of these.
4. When a timeline comment is added to a video source other than the one originally loaded, a screenshot and timestamp is stored in the comment - however the source is *not* changed to match the comment (this is intentional, as the source may not be available).
5. To play any Rich Media content in a PDF file, users of Acrobat and Adobe Reader must manually install the NPAPI (Netscape) version of the Flash Player runtime. Acrobat and Adobe Reader do not include this as part of their installations, and the ActiveX version of Flash Player (used in Internet Explorer) will not work.

Widgets for Adobe® Acrobat® : VideoPlayerX

The Enhanced Video Tool for Adobe Acrobat

Using the listener function

VideoPlayerX build 10.2 and later will look for a document-level JavaScript function named "vpx_listener" and if found, will send event messages to this function. If it's not defined, it won't. The point behind this feature is that in many cases you will want to take some action within the JS API based on events happening in the video (looping, stopping, etc.) but you cannot use the multimedia_registerEventHandler() method until the annotation is activated. If for example you want to automatically deactivate a floating window annotation when playback completes, you can use the vpx_listener() function to watch out for that event, and take the appropriate action.

Of course if your PDF softwre doesn't support JavaScript, or it's disabled, nothing will work.

Events sent to the function

The **vpx_listener()** function should have four parameters (RMA,Page,Pos,Event) and no return value:

```
RMA     name of the Rich Media Annotation (string)
Page    PDF page on which the RMA is located (integer, zero-based)
Pos     the playback position in seconds (float)
Event   the name of the event - such as "playing", "complete", etc. (string)
```

Values of "Event" are:

- buffering
- complete
- connectionError
- disconnected
- loading
- paused
- playing
- resizing
- rewinding
- seeking
- stopped

Examples

Suppose your annotation has a name of "RM2" and has been placed on page 4. You want to disable this annotation (closing the floating window if present) when the video completes playback, but not affect any others in the document. You would define a document-level function as follows:

```
function vpx_listener(r,pg,pos,e) {
  if (r=="RM2" && pg==3 && e=="complete") {
     this.getAnnotRichMedia(pg,r).activated = false;
  }
}
```

Suppose you want to report every status message to the console, and open a website if a remote video fails to load. You would define a document-level function as follows:

```
function vpx_listener(r,pg,pos,e) {
  console.println("vpx_listener: RMA="+r+", page="+pg+", event="+e+", position="+p);
  if (e=="connectionError") app.launchURL("http://www.helpfulwebsite.abc/brokenvideo", true);
}
```

Auto-advancing

Suppose you want to automatically advance to the next page in the PDF file when a video finishes playing (i.e. it reaches the end, not when the user pauses it with the on-screen controls). The annotation should also deactivate itself to free up memory. You would define a document-level function as follows:

```
function vpx_listener(r,pg,pos,e) {
```

```
  if (e=="complete")  {
    this.pageNum++;
    this.getAnnotRichMedia(pg,r).activated = false;
    }
}
```


Widgets for Adobe® Acrobat® : VideoPlayerX

The Enhanced Video Tool for Adobe Acrobat

Using the vpx_init() function

VideoPlayerX build 10.2 and later will look for a document-level JavaScript function named "vpx_init" and if found, will call the function once when the annotation activates. If it's not defined, it won't. The point behind this feature is that in many cases you will want to send one or more video annotations specific setup commands (such as changing skin colors, scale modes, etc.) whenever they activate. As the annotation's Flash stage doesn't exist until it's activated, document-level scripts fired when the PDF opens cannot send those instructions. Placing them in the vpx_init() function you know that the annotation is ready to receive commands, and which annotation it is.

Of course if your PDF softwre doesn't support JavaScript, or it's disabled, nothing will work.

Parameters sent to the function

The **vpx_init()** function should have two parameters (RMA,Page) and no return value:

```
RMA     name of the Rich Media Annotation (string)
Page    PDF page on which the RMA is located (integer, zero-based)
```

Examples

Suppose your annotation has a name of "RM1" and has been placed on page 2. You want to set this annotation to have a red background color, but not affect any others in the document. You would define a document-level function as follows:

```
function vpx_init(r,pg) {
  if (r=="RM1" && pg==1) {
     this.getAnnotRichMedia(pg,r).callAS("multimedia_setStageColor",0xFF0000);
  }
}
```

Suppose you want to set every annotation to use "exactFit" scaling *except* the ones on page 4, which should use "noScale". You would define a document-level function as follows:

```
function vpx_init(r,pg) {
  var _sm = (pg==3)? "noScale":"exactFit";
  this.getAnnotRichMedia(pg,r).callAS("multimedia_setScaleMode",_sm);
}
```

You can use the vpx_init() function to do whatever you want, it doesn't have to relate to controlling the video. Note that in some very rare cases the function may not be fired - if the JSAPI engine takes more than 3 seconds to enter the ready state, VideoPlayerX will assume it's not available and will stop talking. In most cases the JSAPI engine will be ready in less than 500 milliseconds.

