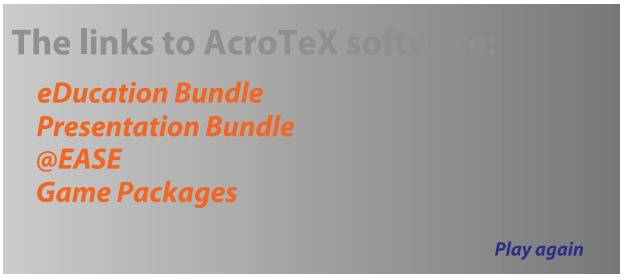
AcroTeX.Net

The `rmannot` Package

Rich Media Annotations

for Acrobat 9 Pro, or later

D. P. Story



Prepared: October 21, 2010

www.acrotex.net
Version 1.0

Table of Contents

1. Introduction

Beginning with version 9, **Adobe Reader** and **Acrobat** contain an embedded **Adobe Flash Player** that will play SWF, FLV, and MP3 files. A new annotation type, called a *rich media annotation*, was developed to manage these media file types in a PDF file.

The `rmannot` package supports the creation of rich media annotations (a `RichMedia` annotation type), and the embedding of SWF, FLV, and MP3 files in a PDF. SWF animations, FLV video, and MP3 sound can then be played within a PDF viewed within version 9 (or later) of **Adobe Reader** or **Acrobat**.[1]

Source material for the creation of this package is the document *Adobe Supplement to the ISO 32000*, June 2008. This document contains the PDF specification—the so called, BaseLevel 1.7, ExtensionLevel 3 specification—of the rich media annotation.

Examples. In addition to the examples that ship with the `rmannot` package, there are numerous examples of `rmannot` on my AcroTeX PDF Blog. I wrote a whole series of articles on the **Rich Media Annotation** (Blogs #1–11) using **AeB Pro** and `rmannot`. Additional examples can be found on my AeB Blog.

2. Requirements

The requirements for your LaTeX system, and well as any other software, is highlighted in this section.

2.1. LaTeX Package Requirements

The following packages, in addition to the standard LaTeX distribution, are required:

1. The `xkeyval` package is used to set up the key-value pairs of the `\rmAnnot` command. Get a recent version.
2. AeB (AcroTeX eDucation Bundle) The most recent version. In particular the `eforms` package and its companion package `insdljs`. The AeB Pro package is recommended. (All the demo files use AeB Pro.)
3. The `graphicxsp` Package. The latest version, I made some slight modifications of this package for `rmannot`. This package allows the embedding of poster graphics for use in the appearances of the annotations when they are not activated.

If you don't have AeB or `graphicxsp`, you can obtain the latest versions from The AeB Home page[2] and the GraphicxSP home page.[3] The installation instructions for AeB must be read very closely as there are certain JavaScript files that must be copied to the correct location on your local hard drive. The AeB Pro package can be obtained from its home page as well.[4] It too has a JavaScript file that needs to be installed, placement

[1] The `rmannot` package was written, in part, to support the AcroTeX Graphing package.
[2] AeB: http://www.math.uakron.edu/~dpstory/webeq.html
[3] GraphicxSP: http://www.math.uakron.edu/~dpstory/graphicxsp.html
[4] AeB Pro: http://www.math.uakron.edu/~dpstory/aeb_pro.html

is important.

2.2. PDF Creator Requirements

The rmannot package supports **Acrobat Distiller 9.0** (or later) as the PDF creator. The document author must have **Acrobat 9.0 Pro** and its companion application **Distiller**. The document author typically uses dvips to produce a PostScript file, which is then distilled to obtain a PDF.

Important: For users of **Acrobat X Pro**, F4V files are supported by the rmannot package, and F4V files are playable in **Adobe Reader X**.

3. Installation

The installation is simple enough. Unzip `rmannot.zip` in a folder that is on your LaTeX search path. Refresh your filename database, if appropriate.

I am perhaps the last one using YandY, but if there is anyone else, there is one other thing to do. The distribution comes with the default poster file for the MP3 file; the name of this file is `ramp3poster.eps` (found in the `graphics` subfolder). For YandY users, this file needs to be copied to a folder on the PSPATH. If you don't know what I'm talking about, follow the steps below.

Open dviwindo, and go to `Preferences > Environment` and choose PSPATH from the drop down menu. Add the path

```
C:\yandy\tex\latex\contrib\rmannot\graphics\\
```

at the end of your PSPATH string.[5] It is important to have the double backslash at the end of the path. This tells the YandY System to search all subfolders for the graphics files. When you are finished, your PSPATH should look something like this:

```
C:\yandy\ps;C:\yandy\tex\latex\contrib\rmannot\graphics\\
```

Be sure to separate these paths by a semicolon.

Important: In recent versions of Acrobat, security restrictions have been put in place to prevent **Distiller** from reading files (the PostScript **file** operator does not work). Fortunately, Distiller has a switch that turns off this particular restriction. To successfully use this package, therefore, you need to run Distiller by using the `-F` command line switch. I personally use the WinEdt application as my text editor,[6] and have defined a Distiller button on my toolbar. The Distiller button executes the following WinEdt macro.

[5] If your YandY System installation is elsewhere, enter that path.
[6] WinEdt home page: www.winedt.com

```
Run(|"c:\Program Files\Adobe\Acrobat 9.0\Acrobat\acrodist.exe" -F "%P\%N.ps"|,
    '%P',0,0,'%N.ps - Distiller',1,1);
```

Note the use of the `-F` switch for `acrodist.exe`. If this package is used to create rich media annotations without the `-F` switch, you typically get the following error message in the Distiller log file

```
%%[ Error: undefinedfilename; OffendingCommand: file ]%%
```

This tells you that either you have not started Distiller with the `-F` command line switch, or Distiller can't find one of the files that the **file** operator was trying to read.

Mac OS Users. The above comments on the `-F` command line switch is for Windows users, Mac OS user must choose the `AllowPSFileOps` user preference, this is located in the `plist`, possibly located at

```
/Users/[User]/Library/Preferences/com.adobe.distiller9.plist
```

You can use Spotlight, the search utility on Mac, to search for `com.adobe.distiller`. This finds the file `com.adobe.distiller9.plist`. Clicking on this find, Spotlight opens `com.adobe.distiller9.plist` in the `plist` editor, see Figure 1. If necessary, click on the arrow next to the Root to expand the choices, then click the up and down arrows at the far right in the `AllowPSFileOps` row to select Yes as the value.

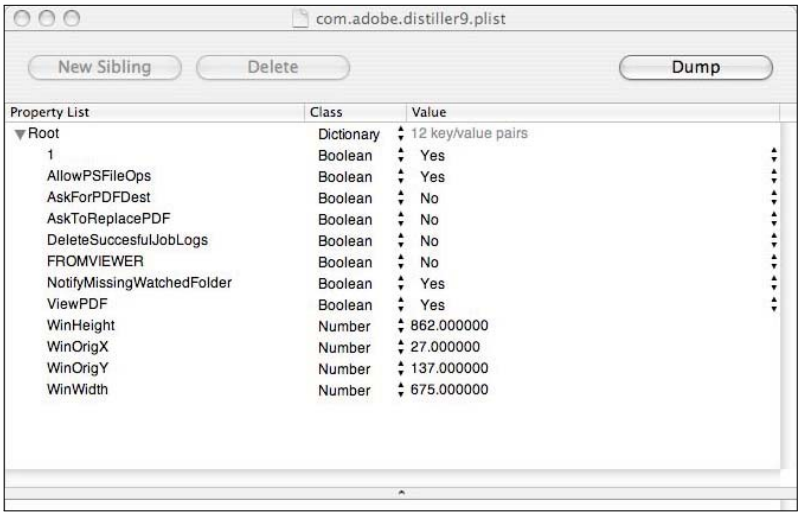


Figure 1: com.adobe.distiller9.plist

4. Setting the Paths and Posters

The paths to SWF/FLV/MP3 files are required to appear in the preamble, and any poster graphics are required to appear in the preamble as well.

4.1. Setting the Paths

There are two types of paths: System paths to resources needed by **Acrobat Distiller**, and media paths to the files used in the document.

System Paths. This package uses **Acrobat Distiller 9.0** (or later), and requires the document author to have **Acrobat 9 Pro**. In the **Acrobat** program folder is a `Multimedia Skins` folder. This folder contains skins (SWF files) used in providing playing controls to FLV video files, and in the `Players` subfolder you will find VideoPlayer.swf and AudioPlayer.swf. The former plays FLV files with an appropriate skin for user controls, the latter plays MP3 files. The document author needs to set these paths to these files, which are passed on to the distiller. For this purpose use `\pathToSkins`. The default definition follows:

```
\pathToSkins{%
    C:/Program Files/Adobe/Acrobat 9.0/Acrobat/Multimedia Skins}
```

This is the path on my WinXP system. The path for the Mac OS may look like

```
\pathToSkins{/Applications/Adobe\ Acrobat\ 9\ Pro/Adobe\ Acrobat\
    Pro.app/Contents/MacOS/Multimedia\ Skins}
```

These paths may differ from platform to platform. Note what the path is to the Multimedia Skins folder. The command `\pathToSkins` also defines the path to the `Players` subfolder. Future releases of **Acrobat** may change the name of the folders, so a `\pathToPlayers` command is also provided; as with `\pathToSkins`, `\pathToPlayers` takes one argument, the path to the players.

☛ The rmannot distribution comes with a `rmannot.cfg` file. In this file, you can set the path to the multimedia skins (`\pathToSkins`) on your system. Read the instructions contained in this file.

Document Media Paths. Each media file (SWF, FLV, MP3) must be declared in the preamble using the `\saveNamedPath` command.

```
\saveNamedPath[<mime_type>]{<name>}{<path>}
```

The first optional argument `<mime_type>` is normally not needed. It is the mime type of the file. Currently, only SWF, FLV and MP3 files are supported, and the extension of the file name is isolated to determine the mime type. The second parameter `<name>` is

a *unique* name that will be used to reference this media file. Finally, <path> is full and absolute path to the media file. The path includes the file name and extension.

For example,

```
\saveNamedPath{mySWF}{C:/myMedia/AcroFlex3_demo.swf}
\saveNamedPath{fishing}{C:/myMedia/100_0239.flv}
\saveNamedPath{summertime}{C:/myMedia/Summertime.mp3}
```

Once the paths are defined in this way, the media files are referenced using their given names. This has a couple of purposes.

1. The names are used to determine if the media file has already been embedded in the document. Though the media clip may be used in several rich media annotations, the `rmannot` attempts to embed a media file only once.
2. The command `\saveNamePath` uses `\hyper@normalise`, of the `hyperref` package, to "sanitize" special characters, so the path may contain characters that normally have special meaning to LaTeX.
3. Defining the path once leads to a consistent reference to the file paths, and reduces the chance of typos.

A brief example to illustrate the use of the names assigned by the `\saveNamedPath` follows:

```
\rmAnnot{200bp}{200bp}{mySWF}
```

See '\rmAnnot and its Options' on page 9 for additional details on the `poster` key and the `\rmAnnot` command.

The above example would use the default poster image to give a visual of the annotation when it is not activated. The next section discusses how to define and implement your own poster image.

Defining a RM Path. The resources (`.flv`, `.swf`, `.mp3` files, for example) for your Flash application my reside on your local computer or in the Internet. As a way of reducing the amount of typing, you can use `\defineRMPath` to define common paths to your resources.

```
\defineRMPath{<\name>}{<path>}
```

The command uses `\hyper@normalise` (of `hyperref`) to "sanitize" the path. The first argument <\name> is the name of the command to be created, and <path> is the path. After the definition <\name> expands to <path>. For example,

```
\defineRMPath{\myRMFiles}{C:/myMedia}
\saveNamedPath{mySWF}{\myRMFiles/AcroFlex3_demo.swf}
\saveNamedPath{fishing}{\myRMFiles/100_0239.flv}
\saveNamedPath{summertime}{\myRMFiles/Summertime.mp3}
```

We first define a path to our resources, then save those paths along with the file names.

You can use \defineRMPath to define URLs as well

```
\defineRMPath{\myRMURLs}{http://www.example.com/~dpspeaker/videos}
```

Now, \myRMURLs points to your common video resources on the Internet.

4.2. Creating Posters

The \rmAnnot command has a poster key that is recognized as part of optional key-value pairs. The use of the poster key is optional, if you do not specify one, one will be generated for you. (More on the default poster appearance is presented below.) The poster image is visible when the rich media annotation is not activated.

To create a poster for your rich media annotation, use a graphics application (Adobe Illustrator, Adobe Photoshop, etc.), and save as an EPS file. Move this file to your source file folder. Let's call this file cool_poster.eps. In the preamble place the command

```
\makePoster{myCP}{cool_poster}
```

The first argument is a *unique name* for the graphic, the second argument is the path name of the graphic (without the extension). The name is used as the value of the poster key.

The command actually has an optional first argument. This argument is passed to the command \includegraphics (of the graphicx package). The general syntax of the command is

```
\makePoster[<options>]{<name>}{<path_to_EPS>}
```

The command uses the graphicxsp package to embed the file in the PDF document. The graphical image can then be used multiple times in many annotations.

The graphic itself should have the same *aspect ratio* as the rich media annotation; this is important if the graphic contains text or images that would get otherwise distorted.

Example,

```
\rmAnnot[poster=myCP]{200bp}{200bp}{mySWF}
```

See '\rmAnnot and its Options' on page 9 for additional discussion of the poster key and \rmAnnot.

Default Poster Image. The rmannot package has default poster appearance. This poster appearance takes one of two forms. If the media file is MP3, an image of the AudioPlayer control bar is used; otherwise it is dynamically generated (with the correct dimensions) using the following PostScript operators:

```
\defaultPoster
{%
    .7529 setgray
    0 0 \this@width\space\this@height\space rectfill
    10 \adj@measure 10 \adj@measure moveto .4 setgray
    /Helvetica \this@height\space 10 div selectfont
    (\rma@posternote) show
}
```

The commands `\this@width` and `\this@height` are the width and height of the annotation. The command `\adj@measure` converts a measurement to a proportion of the smaller of the two measurements `\this@width` and `\this@height`.[7]

Note that, in the above code, some text is generated in the lower left corner of the annotation, the text is `\rma@posternote`. This command is populated by the value of the `posternote` key of the optional argument of `\rmAnnot`. The default value of `posternote` is `AcroTeX Flash` or `AcroTeX Video`, depending on the file type of the media. This can be changed through the `posternote` key.

The default poster itself can be redefined by a document author who is schooled in PostScript things, perhaps if only to change colors, or font, or location of the poster note.

5. `\rmAnnot` and its Options

The `\rmAnnot` command creates a rich media annotation, new to version 9 of **Acrobat**/**Adobe Reader**. Media files (SWF, FLV, or MP3) can be either embedded in the document, or linked via a URL, and played. **Acrobat**/**Adobe Reader** have a built-in Flash player that plays SWF, FLV and MP3 files.

Media files in other formats need to be converted to one of these three supported formats.[8]

5.1. `\rmAnnot` Command

The primary command of this package is `\rmAnnot`, which has four arguments, one optional and three required.

```
\rmAnnot[<options>]{<width>}{<height>}{<name>}
```

[7] The code presented here is a simplified version of the actual code found in `rmannot.dtx`. The definition of the default poster has a number of macros that can be redefined to change the placement of text, the color, size of the font, etc. See `rmannot.dtx` for details.

[8] The new **Acrobat 9 Pro Extended** can convert media files to FLV, but embed the converted file in the PDF, so we cannot really use that re-encoded file with our `rmannot` package. Adobe Flash Video Encoder converts many movie formats to FLV format, which can, in turn, be used in this package. Other utilities may be available as shareware or commercialware.

The <width> and <height> parameters are what they are, the width and height to be used in the rich media annotation. The aspect ratio should be the same as the aspect ratio of the Flash media. The annotation can be resized using either \resizebox or \scalebox of the graphicx package to get the physical dimensions you want.

For MP3 Files. After a careful measurement, the aspect ratio (width/height) of the MP3 AudioPlayer control bar is about 9.6. In some of the demo files, I've been using a width of 268bp and a height of 28bp, and resize the annotation to what is desired. Use 268bp and 28bp for the width and height of an MP3 file, and resize.

The <name> argument references a media file defined by the \saveNamedPath in the preamble.

The <options> are discussed in the subsection that follows.

• \rmAnnot Options

The \rmAnnot command has many key-value pairs that are passed to it through its first optional argument. Most of these key-value pairs correspond to options available through the user interface of **Acrobat**. Below is a listing of the key-values, and a brief description of each.

- name: The name of the annotation. If none is supplied, then aebRM\therm@Cnt is used, where rm@Cnt is a LaTeX counter that is incremented each time \rmAnnot is expanded.
- enabled: The enabled key determines when the annotation is activated, possible values are onclick, pageopen, and pagevisible.
  - onclick: The annotation is activated when the user clicks on the annotation, or is activated through JavaScript.
  - pageopen: The annotation is activated when the page containing the annotation is opened.
  - pagevisible: The annotation is activated when the page containing the annotation becomes visible. (Useful for continuous page mode.)

  The default is onclick.
- deactivated: The enabled key determines when the annotation is activate, possible values are onclick, pageopen, and pagevisible.
  - onclick: The annotation is deactivated by user script or by right-clicking the annotation and choosing Disable Content.
  - pageclose: The annotation is deactivated when the page containing the annotation is closed.
  - pageinvisible: The annotation is deactivated when the page containing the annotation becomes invisible. (Useful for continuous page mode.)

The default is onclick.

- windowed: A Boolean, which if true, the media is played in a floating window. The default is false, the media is played in the annotation on the page. For information on how to set the floating window parameters, see 'Setting the Floating Window Parameters' on page 15.
- url: A Boolean, which if true, the media is to be interpreted as an URL. The default is false, the media is embedded from the local hard drive and embedded in the PDF file.
- borderwidth: The borderwidth determines whether a border is drawn around the annotation when it is activated. Possible values are none, thin, medium, and thick. The default is none.
- poster: The name of a poster graphic created by \makePoster. See the section 'Creating Posters' on page 8 for additional details.
- posternote: When the poster key is not given, the default poster is generated. A short note of text appears in the lower left-corner. The text for that note can be passed to the default poster appearance through posternote. See 'Creating Posters' on page 8 for additional details.
- invisible: A Boolean which, if present, rmannot creates a transparent poster for the RMA. The RMA has not hidden property as form fields do, the best you can do is to give the RMA a transparent poster and place it in an obscure corner of the page, or under a form field. Normally, if invisible is specified, the video content is played in a window (that is, windowed is specified as well).

  Note: The invisible option requires that you distill the document with a job options setting of Standard_transparency, distributed with the graphicxsp package.
- transparentBG: This option is available for SWF files only. Quoting the *Adobe Supplement* document, "A flag that indicates whether the page content is displayed through the transparent areas of the rich media content (where the alpha value is less than 1.0). If true, the rich media artwork is composited over the page content using an alpha channel. If false, the rich media artwork is drawn over an opaque background prior to composition over the page content." The default is false.
- passcontext: A Boolean, if true, passes right-click context to Flash. Should be used only if there is a way of deactivating the annotation, perhaps through JavaScript. Recognized only for SWF files. The default is false.

  SWF file developers can select this option to replace the Acrobat context menu with the context menu of the originating SWF file. When the user right-clicks the SWF file, the available options are from the originating file.

- skin: For playing a FLV file, seven different skins are available for the user to control the video, skin1, skin2, skin3, skin4, skin5, skin6, and skin7. Another possible value is none, for no skin. In the latter case, the media is played when activated, but there is no user interface to control the play. As for the description of each of the skins,
  - skin1: All Controls
  - skin2: Play, Stop, Forward, Rewind, Seek, Mute, and Volume
  - skin3: Play
  - skin4: Play and Mute
  - skin5: Play, Seek, and Mute
  - skin6: Play, Seek, and Stop
  - skin7: Play, Stop, Seek, Mute, and Volume
  - none: No Controls
- skinAutoHide: A Boolean, if true, the skin auto hides. Only valid for FLV files.
- skinBGColor The color of the skin. The value is a color in hex format. The default is 0x5F5F5F. Only valid for FLV files.
- skinBGAlpha: The alpha level of the skin, a number between 0 and 1. The default is 0.75. Only valid for FLV files.
- volume The initial volume level of the video file, a number between 0 (muted) and 1 (max volume). The default is 1.0. Only valid for FLV files.
- speed: Description quoted from the *Adobe Supplement* document. "A positive number specifying the speed to be used when running the animation. A value greater than one shortens the time it takes to play the animation, or effectively speeds up the animation." The default is 1.
- playcount: Description quoted from the *Adobe Supplement* document. "An integer specifying the play count for this animation style. A nonnegative integer represents the number of times the animation is played. A negative integer indicates that the animation is infinitely repeated." The default is -1.
- cuepoints: If the video is encoded with cue points, you can associate a JavaScript action with each. The value of cuepoints is a comma delimited list of cue points. See the paragraph 'On Cue Points' on page 14 for more details.
- resources: Use this key to list all files that are required to run a SWF file. The value of the resources key is a comma delimited list of path names created by the \saveNamedPath command. *The files referenced within this key are embedded in the PDF.* Files that are on the Internet—and are played from the Internet—should not be listed here.
- flashvars: Flash developers can use the flashvars key to add ActionScript variables for the SWF file. See the discussion of *Some Name commands* below.

Some Name commands. Within the optional parameters of the \rmAnnot command, two convenience commands, \Name and \urlName, are defined. They can be used, for example, with the flashvars key.

The \Name command may be used to set the value of a flash variable. \Name has one argument, the symbolic name of a file embedded by \saveNamedPath. The expansion of \Name{<name>} will appear in the Resources tab of the Edit Flash dialog box. For example, if we define myVid as

```
\defineRMPath{\myRMFiles}{C:/acrotex/video}
\saveNamedPath{myVid}{\myRMFiles/assets/myVid.flv}
```

then \Name{myVid} expands to myVid.flv. If the path is grouped with braces, like so,

```
\saveNamedPath{myVid}{\myRMFiles/{assets/myVid.flv}}
```

then \Name{myVid} expands to assets/myVid.flv. This latter form corresponds to adding a directory using the Add Directory button on the Resources tab of the Edit Flash dialog box.

We can then use \Name as follows:

```
\rmAnnot[flashvars={source=\Name{myVid}},
    resources={myVid}]{320bp}{240bp}{mySWF}
```

where mySWF is the name of an SWF application that takes a flash variable named source, the value of the variable is the video to be played.

The \urlName command is designed for resources on the Internet, and which are passed to the SWF application with a flash variable.

```
\defineRMPath{\myRMURLs}{http://www.example.com/~dpspeaker/videos}
\saveNamedPath{myVid}{\myRMURLs/myVid.flv}
```

The expansion of \urlName{myVid} is

```
http://www.example.com/~dpspeaker/videos/myVid.flv
```

We can then use \urlName as follows:

```
\rmAnnot[flashvars={source=\urlName{myVid}
}]{320bp}{240bp}{mySWF}
```

Note that we don't list myVid as a resource, we just pass the URL to mySWF as a flash variable.

Note. The \Name and \urlName commands are defined within the optional parameters of Acrobat form fields created by the eforms package.

On Cue Points. A cue point is any significant moment in time occurring within a video clip. Cue points can be embedded in the FLV using **Adobe Flash Professional**, or some other video encoder.

The value of the `cuepoints` key is a list of cue points data, a "typical example" is

```
\newcommand{\myCuePoints}{%
    {type=nav,name=Chapter1,time=0,action={console.println("Chapter1")}},%
    {type=nav,name=Chapter2,time=1883,action={console.println("Chapter2")}},%
    {type=nav,name=Chapter3,time=5197,action={console.println("Chapter3")}},%
    {type=nav,name=Chapter4,time=6817,action={console.println("Chapter4")}},%
    {type=nav,name=Chapter5,time=9114,action={console.println("Chapter6")}},%
    {type=nav,name=Chapter6,time=12712,action={console.println("Chapter6")}}
}
```

Comments: Having made such a definition, we then say `cuepoints={\myCuePoints}`, note that `\myCuePoints` must be enclosed in braces. Note also in the above example, that the comment character (%) is used after each comma (,) in a line break. Because of the way the argument is initially parsed, these comment characters are needed.

Each of the cue points is a comma-delimited list of key-value pairs; the keys are `type`, `name`, `time`, and `action`. Each of these are briefly described.

- `type`: Possible values for this key are `nav` and `event`, and describes the type of cue point this is.
  - `type=nav`: Navigation cue points enable users to seek to a specified part of a file. Embed Navigation cue points in the FLV stream and FLV metadata packet when the FLV file is encoded.

    Navigation cue points create a keyframe at the specified cue point location, so you can use code to move a video player's playhead to that location. You can set particular points in an FLV file where you might want users to seek. For example, your video might have multiple chapters or segments, and you can control the video by embedding navigation cue points in the video file.[9]
  - `type=event`: Event cue points can also be embedded in your FLV stream and FLV metadata packet when video clip is encoded. You can write code to handle the events that are triggered at specified points during FLV playback.[10]
- `name=<name>`: The name of the cue point
- `time=<time>`: The time in milliseconds the cue point occurs.
- `action=<JS_action>` The JavaScript code that is executed when this cue point is reached.

[9] Taken in part from http://www.peachpit.com/articles/article.aspx?p=663087
[10] Ibid.

• Setting the Floating Window Parameters

When the windowed key is set to true, the rich media annotation appears in a floating window. Use the \setWindowDimPos command to set the dimensions of the window and its positioning.

```
\setWindowDimPos{<key-value pairs>}
```

Command Location: This command may be placed anywhere and will take affect for the next rich media annotation created by \rmAnnot.

Parameter Description: There are a number of key-value pairs for setting the floating window; the default values are normally adequate for most applications.

- width: The width is described by three *key-value pairs*, default, max, and min, measured in default user space units. Default values: default: 288, max:576, min: 72.

  For example, width={default=300,max=600,min=80}.

- height: The height is described by three *key-value pairs*, default, max, and min, measured in default user space units. Default values: default: 216, max:432, min: 72.

  For example, height={default=300,max=600,min=80}.

- position: The position of the floating window is described by four key-value pairs.

  - halign: The halign parameter describes the horizontal alignment of the window. Valid values are near, center and far. The default is far. For languages that read from left-to-right, a value of near refers to the left edge of the viewing window; whereas far refers to the right edge of the viewing window. (For right-to-left reading languages, the description of near and far are reversed.)
  - valign: The valign parameter describes the vertical alignment of the window. Valid values are near, center and far. The default is near.
  - hoffset: The description of hoffset is paraphrased from the *Adobe Supplement* document: The offset from the alignment point specified by the halign key. A positive value for hoffset, when halign is either near or center, offsets the position towards the far direction. A positive value for hoffset, when halign is far, offsets the position towards the near direction. The default is 18.
  - voffset: The description of voffset is paraphrased from the *Adobe Supplement* document: The offset from the alignment point specified by the valign key. A positive value for voffset, when valign is either near or center, offsets the position towards the far direction. A positive value for voffset, when valign is far, offsets the position towards the near direction. The default is 18.

In layman's terms the combination of `halign=far, valign=near` puts the floating window in the upper right corner of the active window of Adobe Reader/Acrobat, assuming a left-to-right reading language. The values of `voffset=18,hoffset=18`, moves the floating window 18 points down and 18 points to the left. That would be its initial position.

Note: This feature, the positioning of the window, never worked in Version 9, but has been implemented for Version 10.

The `\resetWindowDimPos` command can be used to reset the floating window parameters to their default values.

```
\resetWindowDimPos
```

5.2. Examples

In this section, several examples are presented that illustrate the `\rmAnnot` and some of the key-value pairs.

• Posters

The poster is an image that is displayed when the rich media annotation is not activated. If a poster is not specified using the `poster` key, one is supplied for it. Consider the following Flash animation.

Above are two rich media annotations, each running the same SWF file. The one on the left uses the default poster, the one on the right uses a custom poster. In the annotation on the left, you see the default `posternote`, this can be changed using the `posternote` key.

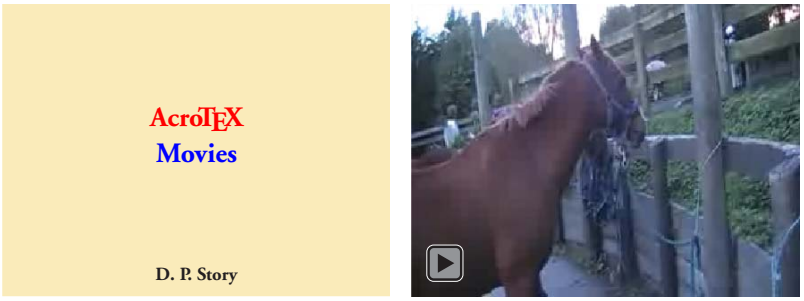
The custom poster was obtained by viewing the SWF file in Adobe Flash Player 9, then printing one of the frames to Adobe PDF, cropping the PDF, then saving the resulting PDF as an EPS file. After you crop the printed image, you can determine its dimensions by moving your mouse to the lower-left corner; the width and height values should appear. Use these in setting up your annotation.

The verbatim listing for the two above annotations is found below.

```
\begin{center}
    \resizebox{!}{.75in}{\rmAnnot{612bp}{265bp}{AcroAd}}\quad
    \resizebox{!}{.75in}{%
        \rmAnnot[poster=AcroAd_poster]{612bp}{265bp}{AcroAd}}
\end{center}
```

The poster AcroAd_poster was defined in the preamble of this document.

Below is the same video, the one on the left is a generic poster created from a LaTeX source file, then saved as an EPS file, the one on the right was obtained from the poster page generated by **Acrobat**. (See the paragraph below, page 17, for details on how this was done.)



The verbatim listing for the two above annotations follows:

```
\resizebox{2in}{!}{%
    \rmAnnot[poster=aebmovie_poster]{209bp}{157bp}{horse1}}\quad
\resizebox{2in}{!}{%
    \rmAnnot[poster=horse1_poster]{209bp}{157bp}{horse1}}
```

Posters and media files are embedded only once, so using the same poster and/or media file multiple times does not increase the file size significantly.

For MP3 files, the default poster is an EPS file that is an image of the player control bar, the example below shows the MP3 poster and audio player.

The code for the above annotation follows:

```
\resizebox{!}{14bp}{\rmAnnot{268bp}{28bp}{trek}}
```

A custom poster can be inserted using the poster key, as usual.

The Acrobat Pro generated poster. To acquire the same poster image that **Acrobat** generates, use the following steps:

1. Open **Acrobat**
2. Drag and drop your SWF or FLV file onto an empty **Acrobat** window
3. Press **Ctrl-P**, or select File > Print

4. Select **Adobe PDF** as the printer
5. Select **Choose paper source by PDF page size**
6. Select **Use custom paper size when needed**
7. Press **OK**
8. A new PDF should be created, and it should be the same size as the poster image
9. Choose File > Save As, select `Encapsulated PostScript (*.eps)` as the **Save as type**
10. Press **Save**, and save to an appropriate folder.

• Skin Options

When a FLV video file is used, the video is played by the VideoPlayer.swf and uses one of the seven standard skins. Customizing information is actually passed using FlashVars. (For FLV files, the user does not have access to the FlashVars, the application, in this case, this package, uses the FlashVars.) Customizing options include a choice of skin, setting the auto hide flag, a choice of the color of the skin, setting the opacity of the skin and setting the initial volume level. The following illustrates some of the options on a short FLV video with a horse theme.

The video on the left shows the default settings (default skin, skin alpha, volume level, etc.), while the same video on the right uses skin6, with skin color of `0xFF0000` (red) and skin alpha level set to 0.25.

Note, click on the AcroTEX logo to play an MP3 file.

5.3. Third-party Video Players

When you play an FLV file, the SWF file `VideoPlayer.swf` is embedded in the PDF. It is `VideoPlayer.swf` that plays the FLV file. It is this SWF file that allows us to customize the look of the RMA, what skin to use, skin color, skin opacity, value, speed, and so on.

The `VideoPlayer.swf` file, which is shipped with Acrobat Pro, version 9 or later, lacks several useful features, among these are the ability to play more than one video in the same rich media annotation (RMA).

In the past year, there have been two extensions to Adobe's `VideoPlayer.swf`:

- `VideoPlayerPlus.swf` is available from Joel Geraci's web site The PDF Developer Junkie Blog. Joel is a guru at Adobe. Extended features are in the form of additional JavaScript API to player more than one video in an RMA, change skins, change skin color, and a few others. Full documentation can be found on the reference blog page.

  I have extensively tested the `VideoPlayerPlus.swf`, and it seems to work as advertised, no problem.

- `VideoPlayerX.swf` is another extension to the video player shipped by Adobe. This one is being developed by **UVSAR**. Full documentation can be found on this page. This player offers a different set of JavaScript APIs, with some overlap with Joel's **PlayerPlus** widget.

Installation of third-party players. If you want to use either or both of these video players, download them from the appropriate web site:

- VideoPlayerPlus.swf
- VideoPlayerX.swf

Joel's widget is already named `VideoPlayerPlus.swf`, but the one from **UVSAR** is named `VideoPlayer.swf`. Rename **UVSAR**'s widget to `VideoPlayerX.swf`. Place both `VideoPlayerPlus.swf` and `VideoPlayerX.swf` into the same folder that contains Adobe's `VideoPlayer.swf`. This is where the `rmannot` package will look for them.

Once you have installed one or both widgets, `rmannot` can use them. If you want to use either of these video players, make one of the following declarations in the preamble

```
\useVideoPlayerPlus
\useVideoPlayerX
```

Choose the player that gives you the feature you most want to use.

If both are placed in the preamble, only the first one will be recognized. No declaration at all means you are using the Adobe's `VideoPlayer.swf`.

Examples. In time, multiple example files will appear at the AeB Blog to illustrate each of these players.

• JavaScript/ActionScript API for Video Players

Normally, we use \rmAnnot to create a RMA to play a FLV (or SWF or MP3) without any controls. The user clicks on the RMA and the media content plays. For FLV files, a skin may be provided to control over the movie once the RMA becomes activated. For a fancier presentation, you might want to create control buttons to control the movie; to do that, you need to use the JavaScript API for the RMA.

In this section we document the JavaScript API for RMA. The resources for this section are the JavaScript for Acrobat API Reference, The PDF Developer Junkie Blog, and **UVSAR**.

The basic methodology for passing a command to the the video player:

1. **Get the RMA object.** To do this use either the `Doc.getAnnotRichMedia()` or `Doc.getAnnotsRichMedia()` methods. Note that in the latter method the word `Annots` is plural, the plural form distinguishes these to methods from each other. The former gets a single RMA object, while the latter returns an array of RMA objects. For work with rmannot, I prefer the use of `Doc.getAnnotRichMedia()`.

   `Doc.getAnnotRichMedia()` takes two arguments, the first is the page number, and second is the name (a string) of the annot. For example

   ```
   var rma = this.getAnnotRichMedia(this.pageNum, "myCoolRMA");
   ```

   The first argument is normally `this.pageNum`, which is a JavaScript property referring to the current page.

2. **Activate the RMA.** Use the `RMA.activated` property, a Boolean:

   ```
   rma.activated=true;
   ```

   You can, as an alternative say, `if(!rma.activated) rma.activated=true;`

3. **Make the call(s).** Use the `callAS` method of the RMA object. For example, if you want to play the video, you might say,

   ```
   rma.callAS("multimedia_play");
   ```

Putting these lines together to play media, we have

```
var rma = this.getAnnotRichMedia(this.pageNum, "myCoolRMA");
if(!rma.activated) rma.activated=true;
rma.callAS("multimedia_play");
```

Those are the basics of making a call over the “bridge” to the video player widget. In the rest of the section, we concentrate on the JavaScript APIs, the third line above `rma.callAS("multimedia_play");`. The first argument of the `callAS` method is a

string which n names the method to use. Note that this first argument is a string. Additional argument may be used if the multimedia method requires them.

The Scripting Bridge between JavaScript and ActionScript. When a JavaScript method, such as `rma.callAS("multimedia_play")`, is executed on the PDF side, the specified ActionScript function `multimedia_play()` is executed in the SWF widget (for example, in `VideoPlayer.swf`). The `callAS` communicates across what is called the "scripting bridge" to the ActionScript engine. For more information on the scripting bridge, see the AcroTEX PDF Blog, blogs #1–#11.

• Core API

The following methods are defined for all three players. The first argument of the `callAS` method is a string, which names the (ActionScript) method to use in the video player widget. The `rmannot` package defines some convenience commands to give the user a consistent experience between video players (**VideoPlayer**, **VideoPlayerPlus**, **VideoPlayerX**).

Method/Description	Command
`multimedia_play():void` Play the video or sound clip from the current location	`\mmPlay`
`multimedia_pause():void` Pause playback of the current media	`\mmPause`
`multimedia_rewind():void` Rewind the media clip to the beginning. This method does not pause the clip.	`\mmRewind`
`multimedia_nextCuePoint():void` Move the play head to the next cue (chapter) point	`\mmNextCuePoint`
`multimedia_prevCuePoint():void` Move the play head to the previous (chapter) point	`\mmPrevCuePoint`
`multimedia_seek(time:Number):void` Move the play location to an offset of time from the beginning of the media, where time is measured in seconds.	`\mmSeek`
`multimedia_mute():void` Mute the audio of the media	`\mmMute`
`multimedia_volume(volume:Number):void` Set the volume level. The volume is a number between 0 and 1 inclusive. A value of 0 mutes the audio, while a volume of 1 sets the volume level to the maximum level.	`\mmVolume`

Examples of usage,

```
var rma = this.getAnnotRichMedia(this.pageNum, "myCoolRMA");
if(!rma.activated) rma.activated=true;
rma.callAS(\mmVolume, .5);  // half-volume
rma.callAS(\mmPlay);        // and play it
```

• API of VideoPlayerPlus

The **VideoPlayerPlus** supports all the functions of the core API, and adds four more functions.

Method/Description	Command
`multimedia_source(path:string):void` Sets a new source file for the video. The video can either be embedded resource or a URL to streaming content.	`\mmSource`
`multimedia_skin(path:string):void` Sets a new skin file to be used by the player. This should be an embedded resource.	`\mmSkin`
`multimedia_skinBackgroundColor(color:uint):void` Sets a new background color for the player skin in the form of `0xRRGGBB`.	`\mmSkinColor`
`multimedia_skinAutoHide(state:boolean):void` Sets the auto hide behavior for the player bar. [`true` or `false`]	`\mmSkinAutoHide`

Examples of usage,

```
    var rma = this.getAnnotRichMedia(this.pageNum, "myCoolRMA");
    if(!rma.activated) rma.activated=true;
    // use embedded video as source
    rma.callAS(\mmSource, "myVideo.flv";
    // use video on web as source
//  rma.callAS(\mmSource, "http://www.example.com/myCool.flv");
    rma.callAS(\mmPlay);        // and play it
```

• API of VideoPlayerX

The **VideoPlayerX** redefines many of the core API, which returned void, to methods that return meaningful information. It also adds many new methods, with some overlap with **VideoPlayerPlus**.

In the table below, the functions marked with an '∗' are also core functions that have been re-defined to have a return value.

Method/Description	Command
`multimedia_pause():Number*` Pause playback of the current media. Returns on success: Playhead time in seconds	`\mmPause`
`multimedia_mute():Number*` Mute the audio of the media Returns on success: Previous volume setting.	`\mmMute`
`multimedia_volume(volume:Number):Number*`	`\mmVolume`

Method/Description	Command
Set the volume level. The volume is a number between 0 and 1 inclusive. A value of 0 mutes the audio, while a volume of 1 sets the volume level to the maximum level. Returns on success: Previous volume setting.	
`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.	`\mmSeekCuePoint`
`multimedia_setSource(url:String):String` Sets the source for the video (a URL or a local file reference). 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.	`\mmSource`
`multimedia_setSkin(skinName:String):void` Sets a new skin file to be used by the player. This should be an embedded resource.	`\mmSkin`
`multimedia_setSkinColor(color:uint):uint` Sets a new background color for the player skin in the form of `0xRRGGBB`.	`\mmSkinColor`
`multimedia_setSkinAlpha(alpha:uint):uint` Sets the background alpha for the player skin (will only take effect where the skin supports alpha changes). Returns on success: Previous alpha value.	`\mmSkinAlpha`
`multimedia_useLocal(isLocal:boolean):String` Switches to the local source if `isLocal` is set to `true`, or to the remote source if `isLocal` is `false`. Returns on success: source filename/URL in string format Returns on error: `"NOT AVAILABLE"`.	`\mmUseLocal`

There are considerably more functions that are not listed here. For a full list, go to the page VideoPlayerX: Enhanced Video Tool for Adobe Acrobat on the **UVSAR** web site.

Examples of usage,

```
var rma = this.getAnnotRichMedia(this.pageNum, "myCoolRMA");
if(!rma.activated) rma.activated=true;
// use embedded video as source
rma.callAS(\mmSource, "myVideo.flv";
```

```
    // use video on web as source
//  rma.callAS(\mmSource, "http://www.example.com/myCool.flv");
    rma.callAS(\mmPlay);        // and play it
```

Note that this is the same example as for **VideoPlayerPlus**, the function call for setting the source has a different name, but using convenience command `\mmSource` allows you to use the same code for "PlayerPlus" and for "PlayerX."

• Methods shared by all Video Players

The three players (**VideoPlayer**, **VideoPlayerPlus**, and **VideoPlayerX**) all share the core API. Beyond the core API, extended API supported by **VideoPlayerPlus** and **VideoPlayerX** are different, but they do have some overlap. Though there is overlap, the common methods may have different names, this is one of the main reasons for the convenience commands. The command, `\mmSource`, for example, expands to the string `"multimedia_source"` for **VideoPlayerPlus**, but for **VideoPlayerX** expands to the string `"multimedia_setSource"`.

Extended API Overlap: `\mmSource`, `\mmSkin`, and `\mmSkinColor`.

For either player (**VideoPlayerPlus** or **VideoPlayerX**), you can dynamically load in a new source file (either local or remote), designate the skin and skin color.

Examples. Again, over time, I plan posting several example files to illustrate **VideoPlayerPlus** or **VideoPlayerX** and their capabilities. Keep your browser set to the AeB Blog.

That's all for now, I simply must get back to my retirement. 🏖