AcroTEX.Net

The rmannot Package

Rich Media Annotations

for Acrobat DC

D. P. Story

Prepared: April 27, 2021

www.acrotex.net
Version v2.2.1, 2021/04/21

Table of Contents

1 Introduction **3**

2 Requirements **3**
2.1 LaTeX Package Requirements 4
2.2 PDF Creator Requirements 4
• rmannot and non-Distiller workflows 5
2.3 Supported Media Formats 5
• Supported video formats 5
• Supported audio format 5
• Supported 3D formats 5
2.4 Comments on Distiller 6

3 Setting the Paths and Posters **7**
3.1 Setting the Paths 7
3.2 Creating Posters 10

4 \rmAnnot and its Options **12**
4.1 \rmAnnot Command 12
• \rmAnnot Options 13
• Setting the Floating Window Parameters 16
4.2 Examples 17
• Posters 17
• Additional remarks on MP3 19
4.3 \rmAnnot and 3D 20

1. Introduction

JavaScript bridge dead

After December 2020, PDF content that uses Flash Player will no longer be supported by Acrobat DC/Reader DC (AA/AR), this includes all files of type FLV and SWF. As a result, Rich Media Annotations (**RMA**) has changed, its "richness" is now "plain-ness"; of particular importance, the JavaScript Bridge has collapsed, there is no control of the media using JavaScript. The good news is the rmannot package remains functional! Supported file types are,

- **Video:** MOV, MP4, M4V, 3GP, 3G2
- **Audio:** MP3
- **3D:** U3D, PRC

This documentation describes the revised package.

Legacy documentation prior to the EOL of Flash player support. Should you have a version of Acrobat prior to the EOL of Flash player, refer to the legacy documentation rmannot_man-flash.pdf.[1] None of the code of rmannot has been removed, so you can still incorporate FLV/SWF files into your document, but such media may not run on your computer; it pays to keep AA/AR XI or earlier.

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.

At the time of this writing. Recent versions of AA/AR DC offer the option of using Flash content or of using the new multimedia methods. This option is found under Edit > Preferences, select the category 3D & Multimedia Options in the left panel, as shown in Figure 1. At the top of the right-hand panel observe the checkbox item Use Flash Player for playing 3D and multimedia content. By clearing this checkbox, you can experience the new multimedia player. After December 2020, these options will probably be removed.

Examples. In addition to the examples that ship with the rmannot package, there are numerous examples of rmannot on my AcroTeX Blog (having tag *rmannot-package*). There is also a whole series of articles on the **Rich Media Annotation** using AeB Pro and rmannot.[2]

2. Requirements

The requirements for your LaTeX system are highlighted in this section.

[1] Flash player is still supported by Acrobat XI/Adobe Reader XI or prior. Any version of Acrobat DC/Adobe Reader DC, however, may not play Flash content, even though it is an older DC version because of any updates received.

[2] Articles that use FLV or SWF files, or uses the "JavaScript bridge" will not work in recent versions of Acrobat DC or Acrobat Reader DC after EOL of Flash player.

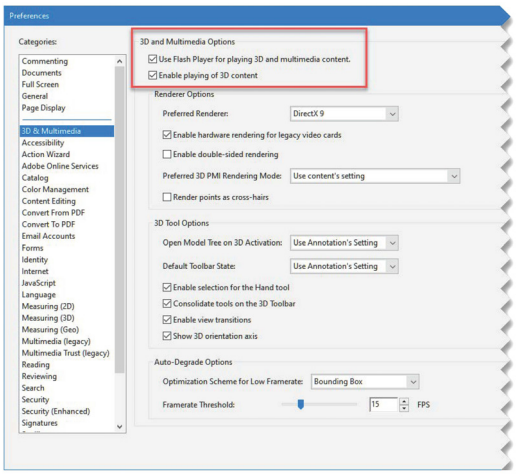


Figure 1: 3D & Multimedia Options

2.1. LaTeX Package Requirements

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

1. 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.) Get it at ctan.org/pkg/acrotex.

2. 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. Get it at ctan.org/pkg/graphicxsp.

3. (Recommended) Many of the demo files use AeB Pro (ctan.org/pkg/aeb-pro) is a recommended addition to your AcroTeX collection.

The installation instructions for AeB and AeB Pro must be read very closely as there are certain JavaScript files that must be copied to the correct location on your local hard drive.

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 (or later) and its companion application Distiller.[3] The document author typically uses dvips to produce a Postscript file, which is then distilled to obtain a PDF.

Protected Mode

If you are using Acrobat DC to build rmannot, it is necessary *to clear the checkbox* Enable Protected Mode at startup, this checkbox is found under the menu Edit > Prefer-

[3] Current Acrobat is Acrobat DC

ences. In the Preferences dialog box, select Security (Enhanced) category from the left panel; the targeted checkbox is at the top line on the right panel. Refer to Figure 2 for a visual.

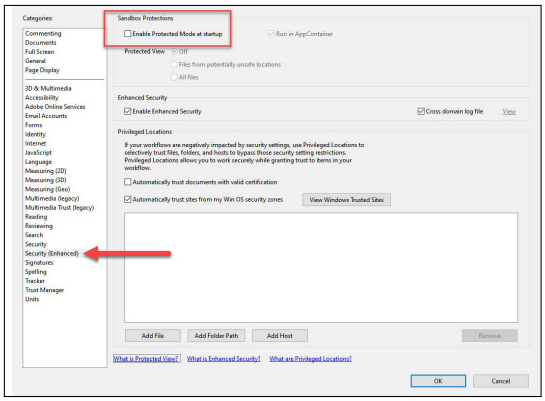


Figure 2: Security (Enhanced) Preferences

• rmannot and non-Distiller workflows

With rmannot dated 2021/04/21 or later, a non-Distiller workflow can be utilized to compile and build a PDF; however, any Rich Media annotation created by `\rmAnnot` is non-functional and is represented by a frame box with the caption Distiller required. This allows the document author to use his/her favorite PDF creator and favorite PDF previewer to develop the document. To build the document to be published, Acrobat/Distiller are required at the tail end of the workflow tex -> dvi -> ps -> pdf.

2.3. Supported Media Formats

• Supported video formats

The resource for video formats is Supported file formats Acrobat and Reader, see the sections **Video formats (Acrobat Pro DC)**, **Video formats (Acrobat X Pro)** and **Video formats (Acrobat 9 Pro and Pro Extended)**. The rmannot package generally supports all formats listed there that have a 'Yes' in the column labeled **Direct placement without transcoding**; in particular, rmannot supports ~~SWF~~, ~~FLV~~, MOV, MP4, M4V, 3GP, 3G2, and MP3 files. Some of these are not supported by version 9. ~~For greatest compatibility, use SWF, FLV (or F4V, Version 9.2 or later).~~

• Supported audio format

The resource for audio formats is Supported file formats Acrobat and Reader, see the section **Audio formats (Acrobat)**. For assured compatibility, use MP3 files for audio.

• Supported 3D formats

The resource for audio formats is Supported file formats Acrobat and Reader, see the section **2D and 3D formats**. Acrobat and this package support filetypes U3D and PRC.

2.4. Comments on Distiller

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,[4] and use the GSView button on the toolbar to pass the PS file to Distiller with the `-F` switch. toolbar. Edit `GSView.edt` to read,

```
Run(|%$('GSView'); %1 -F "%P\%N.ps"|,'%P',0,0,'%N.ps - GSview',1,1);
```

Note the use of the `-F` switch following `%1`. The argument `%1` is the path to the "GSView" application; this is redefined, as shown in Figure 3.

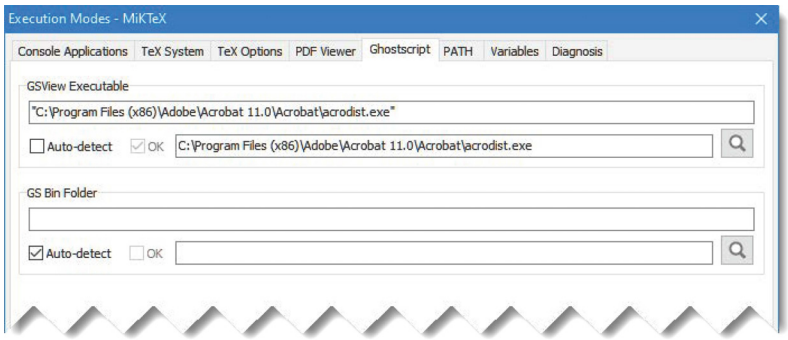


Figure 3: Options > Execution Modes > Ghostscript

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 are for Windows OS users, Mac OS users 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 4. 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.

[4] WinEdt home page: www.winedt.com

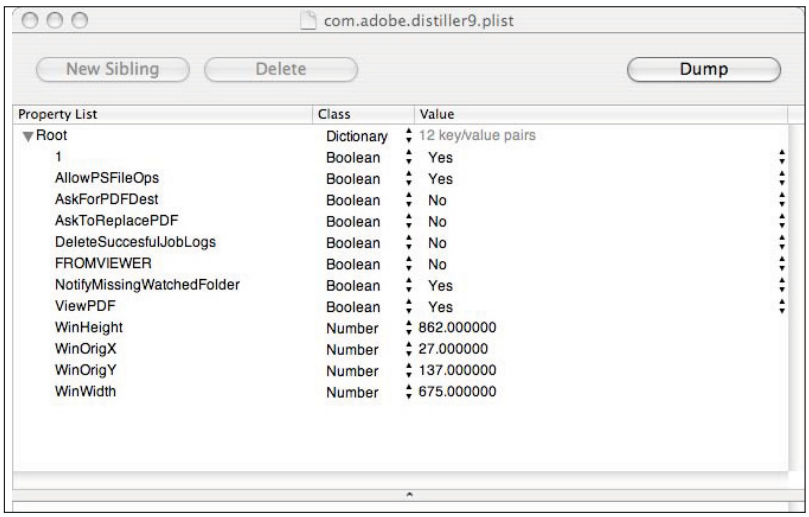


Figure 4: com.adobe.distiller9.plist

For Acrobat DC users. Recent versions of Acrobat Distiller DC have a switch labeled Trust all files opened via Acrobat Distiller under File > Preferences, as shown in Figure 5 on page 8. When this checkbox is checked, there is no need for the -F switch.

3. Setting the Paths and Posters

The paths to the supported media files are required to appear in the preamble, and any poster graphics are required to appear in the preamble as well.

3.1. 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 DC, which requires the document author to have Acrobat DC.[5] In the Acrobat program folder is a `Multimedia Skins` folder. ~~This folder contains the 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 the path to these files, which are passed on to the distiller. This is easily done using the `\AcroVer` command.

```
\AcroVer[win=⟨32|64⟩|mac]{⟨version⟩}
```

Place the command in the preamble or in the `rmannot.cfg` configuration. The optional argument (`win` or `mac`) indicates the platform used to build the document; the `win`

[5] In the post-FLV/SWF era, in theory versions of Acrobat built prior to the EOL of Flash player should work correctly for this package.

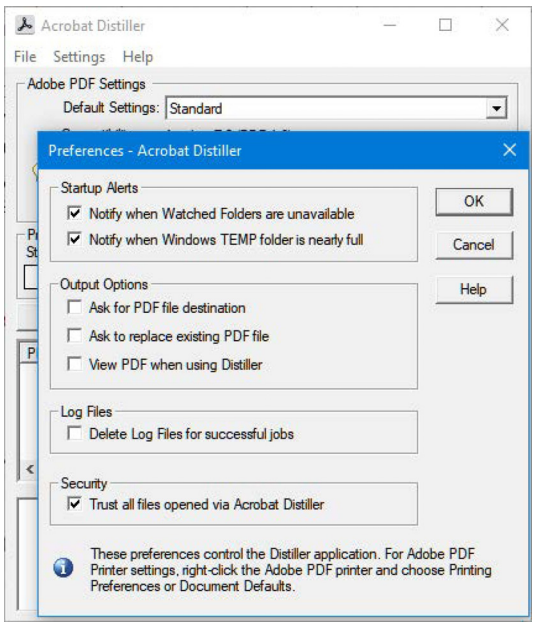


Figure 5: Acrobat Distiller: File > Preferences

option now takes one of two value, 32 or 64. This change is needed as Adobe rolls out its 64-bit versions of Acrobat and Adobe Reader. The required argument is the version of Acrobat you are using to build your **RMA** document. When no optional argument is passed, `win` is assumed (Windows OS). Possible values for ⟨*version*⟩ are DC, a year (2015 or later), or a version number, such as 9, 10, or 11.[6] The default is `\AcroVer{DC}`, which is equivalent to `\AcroVer[win=32]{DC}`.

The rmannot package, based on the information passed to it by `\AcroVer`, builds the appropriated path and passes this path to the `\pathToSkins` command as its argument. Should the path be proven to be incorrect, you can hunt down the correct path and directly enter it in the preamble, or in the `rmannot.cfg` configuration file. For version XI (version 11) of Acrobat, for example, the path is,

```
\pathToSkins{C:/Program Files (x86)/Adobe/%
    Acrobat 11.0/Acrobat/Multimedia Skins}
```

The path for the Mac OS may look like this,

```
\pathToSkins{/Applications/Adobe\ Acrobat\ XI\ Pro/Adobe\ Acrobat\
    Pro.app/Contents/Resources/Multimedia\ Skins}
```

These paths differ from platform to platform and `\AcroVer` tries to take all platforms and versions into consideration.

[6] A value of `Beta` is also recognized, for those in the Beta Program of Acrobat.

☛ The rmannot distribution comes with a `rmannot.cfg` file. In this file, you can place the `\AcroVer` command with its appropriate arguments for your platform and version of Acrobat. Remember, if you update your Acrobat, update also the ⟨*version*⟩ argument of `\AcroVer`.

Document Media Paths. Each media file must be declared in the preamble using the `\saveNamedPath` command.

```
\saveNamedPath[⟨mime_type⟩]{⟨name⟩}{⟨path⟩}
```

The optional argument ⟨*mime_type*⟩ is normally not needed. Only media with the extensions listed in Section 2.3 are supported. 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{calliope}{C:/myMedia/calliope.mp4}
\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{1280bp}{720bp}{calliope}
```

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 rich media resources for your **RMA** may reside on your local computer or on the Internet. As a way of reducing the amount of typing, you can use `\defineRMPath` to define common paths to your resources.

```
\defineRMPath{⟨\cmd⟩}{⟨path⟩}
```

The command uses `\hyper@normalise` (of `hyperref`) to "sanitize" the path. The first argument ⟨*\cmd*⟩ is the command to be created, and ⟨*path*⟩ is the path to the media folder. After the definition, the command ⟨*\cmd*⟩ expands to ⟨*path*⟩. For example,

```
\defineRMPath{\myRMFiles}{C:/myMedia}
\saveNamedPath{calliope}{\myRMFiles/calliope.mp4}
\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.

3.2. Creating Posters

A *poster* is an EPS graphics file that covers the **RMA** when the annotation is not activated. 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. You can also create a single page PDF from a TEX source file, as this package does. 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.For example,

```
\rmAnnot[poster=myCP]{1280bp}{720bp}{calliope}
```

See '`\rmAnnot` and its Options' on page 12 for additional discussion of the `poster` key and `\rmAnnot`.

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.

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
{%
  \rma@ps@bg@setcolor
  0 0 \this@width\space\this@height\space rectfill
  \rma@ps@txt@x\adj@measure\rma@ps@txt@y\adj@measure moveto
  \rma@ps@txt@setcolor/\rma@ps@font
  \rma@ps@relfontsize\rma@ps@fontsize selectfont
  \rma@ps@msg
}
```

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.

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' or 'AcroTeX MP3', 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.

The \setPosterProps command. The parameters of the above display can be set with the \setPosterProps command.

```
\setPosterProps
{%
  color=⟨color⟩,
  xPos=⟨num⟩,
  yPos=⟨num⟩,
  textColor=⟨color⟩,
  relTextSize=⟨num⟩,
  textSize=⟨num⟩,
  textFont=⟨font-name⟩
}
```

Default settings

```
color=.7529 setgray
xPos=10
yPos=10
textColor=.4 setgray
relTextSize=10
textSize=10
textFont=Helvetica
```

The defaults are shown to the right of the display frame box. All values are either numbers or Postscipt code.

color=⟨*color*⟩ The background color of the default poster. The ⟨*color*⟩ specification is a Postscript color operator and its arguments.

`xPos=`⟨*num*⟩ The horizontal starting position (from the lower-left corner) of the `posternote` text. This is a relative value; ⟨*num*⟩ is measured as percentage of the smaller of the two dimensions of the annotation.

`yPos=`⟨*num*⟩ The vertical starting position (up from the lower-left corner) of the `posternote` text. This is a relative value; ⟨*num*⟩ is measured as percentage of the smaller of the two dimensions of the annotation.

`textColor=`⟨*color*⟩ The text color (of the poster note) of the default poster. The ⟨*color*⟩ specification is a Postscript color operator and its arguments.

`relTextSize=`⟨*num*⟩ The size of the text (of the poster note); the value of ⟨*num*⟩ is interpreted as a percentage of the smaller of the two dimensions of the annot. (The text size is resized when the annot is resized.)

`textSize=`⟨*num*⟩ The size of the text of the poster note.

`textFont=`⟨*font-name*⟩ The name of the Postscript font to be use to display the poster note. The default is `Helvetica`.

4. \rmAnnot and its Options

The `\rmAnnot` command creates a rich media annotation (**RMA**), an object first introduced in Version 9 of Acrobat/Adobe Reader (AA/AR), through which media files are played. Media files can be either embedded in the document or linked via a URL.

Media files in other formats need to be converted to one of the supported formats, refer to Section 2.3 for a list of supported media formats.

4.1. \rmAnnot Command

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

```
\rmAnnot[⟨options⟩]{⟨width⟩}{⟨height⟩}{⟨name⟩}
```

The command creates a **RMA** based on the ⟨*name*⟩ and conforming to the dimensions (⟨*width*⟩ and ⟨*height*⟩) and ⟨*options*⟩ specified.

On the dimensions. 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 media media. The annotation can be resized using the `width`, `height`, or `scale` keys, described below.

Acquiring the dimensions. The dimensions can (usually) be obtained from the properties of the media file, as displayed by the operating system. In File Explorer of Windows OS, open the folder containing the media file in question, select the media file of interest, right-click on it and select Properties from the dropdown menu. The media file properties dialog box appear, choose the Details tab, as shown in Figure 6, you can obtain the dimensions of the media from the Frame width and Frame height entries.

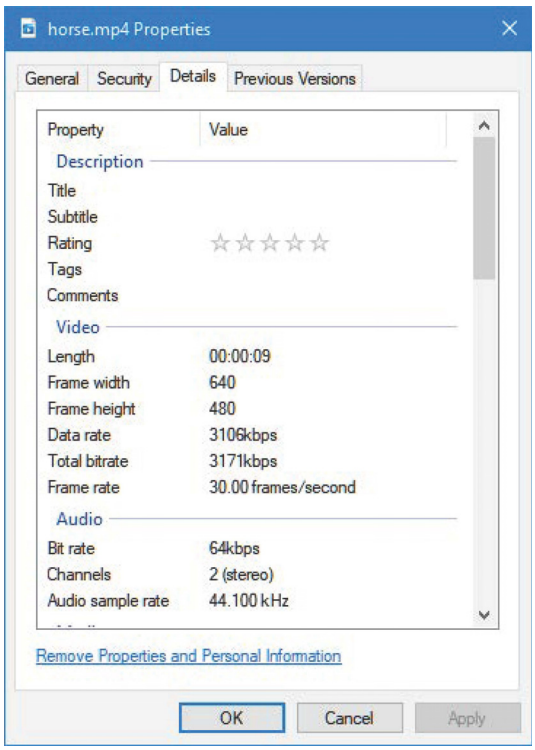


Figure 6: Media file Properties

\cntrlbrWd
\cntrlbrHt

Dimensions for MP3 Files. A special legacy poster can be used. The dimensions of the legacy poster are \cntrlbrWd by \cntrlbrHt. The legacy poster appears when the poster key is not specified. Refer to the named paragraph **Default MP3 poster** on page 18 and to the subsection "Additional remarks on MP3" on page 19 for more information on MP3 files.

The other arguments. 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=⟨*name*⟩ The value, ⟨*name*⟩, is the name of the annotation. If none is supplied, a name of aebRM\therm@Cnt is used, where rm@Cnt is a LaTeX counter that is incremented each time \rmAnnot is expanded. The value of the name key is used primarily for JavaScript purposes. For example, the code

```
var rma=this.getAnnotRichMedia({nPage:this.pageNum,cName:⟨name⟩});
```

acquires the object for the **RMA** on the current page with a name of ⟨*name*⟩.

`enabled=⟨value⟩` 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=⟨value⟩` The `deactivated` key determines when the annotation is deactivated, possible values are `onclick`, `pageclose`, and `pageinvisible`.

`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=⟨true|false⟩` 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 16.

`url=⟨true|false⟩` 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 within the PDF file.

To stream media, it is necessary to clear the checkbox Enable Protected Mode at startup, this checkbox is found under the menu Edit > Preferences. In the Preferences dialog box, select Security (Enhanced) category from the left panel; the targeted checkbox is the top line on the right panel. Refer to Figure 2 for a visual.

End-users For end-users viewing the streaming media from Adobe Reader DC, the user also needs to clear the Enable Protected Mode at startup as well.

After Dec. 2020 not obeyed `borderwidth=⟨none|thin|medium|thick⟩` The value of the `borderwidth` key determines whether a border is drawn around the annotation when it is activated. The default is none.

poster=⟨*name*⟩ The name of a poster graphic created by \makePoster. See the section 'Creating Posters' on page 10 for additional details.

posternote=⟨*text*⟩ 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 10 for additional details.

New: defaultposter 2020/08/21

defaultposter=⟨true|false⟩ A Boolean, which if true, replaces the default MP3 control bar with the default poster image as described in the paragraph **Default Poster Image** on page 11. The default is true. This key is obeyed only for MP3 files and when the poster key *is not* specified.

After Dec. 2020, this key is not useful

invisible=⟨true|false⟩ 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, the windowed option is specified) and controlled by JavaScript methods.

Note: The invisible option requires that you distill the document with a job options setting of Standard_transparency, distributed with the graphicxsp package.

New: all key 2020/08/21

skin=⟨*value*⟩ For playing a video file, ~~seven~~two different skins are available for the user to control the video, these are all, ~~skin1, skin2, skin3, skin4, skin5, skin6, skin7~~, and none. 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,

all All Controls

~~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~~R

none No Controls

The next three keys resize the **RMA** in the recommended way. Only *one key* is recognized for any \rmAnnot command. They are examined in the order of width, height, and scale.

width=⟨*length*⟩ Resizes the **RMA** to a width of ⟨*length*⟩; the aspect ratio of the annot is preserved. For example, the following

```
\rmAnnot[width=.5\linewidth]{1280bp}{720bp}{calliope}
```

resizes so the width of the annot is `.5\linewidth`.

`height=`⟨*length*⟩ Resizes the **RMA** to a height of ⟨*length*⟩; the aspect ratio of the annot is preserved.

`scale=`⟨*pos-num*⟩ Rescales the **RMA** by a scale factor of ⟨*pos-num*⟩; eg, `scale=.5`.

• 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{⟨KV-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 (⟨*KV-pairs*⟩) for setting the floating window; the default values are normally adequate for most applications.

`width=`⟨*KV-pairs*⟩ The width is described by three *key-value pairs*, `default`, `max`, and `min`, measured in default user space units. The ⟨*KV-pairs*⟩ have the form ⟨*key*⟩=⟨*value*⟩.

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

Default values: `default=288`, `max=576`, `min=72`.

`height=`⟨*KV-pairs*⟩ The height is described by three *key-value pairs*, `default`, `max`, and `min`, measured in default user space units. The ⟨*KV-pairs*⟩ have the form ⟨*key*⟩=⟨*value*⟩.

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

Default values: `default=216`, `max=432`, `min=72`.

`position=`⟨`halign|valign|hoffset|voffset`⟩ The position of the floating window is described by four key-value pairs.

`halign=`⟨`near|center|far`⟩ The `halign` 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=`⟨`near|center|far`⟩ The `valign` parameter describes the vertical alignment of the window. Valid values are `near`, `center` and `far`. The default is `near`.

`hoffset=`⟨*num*⟩ 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=`⟨*num*⟩ 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.

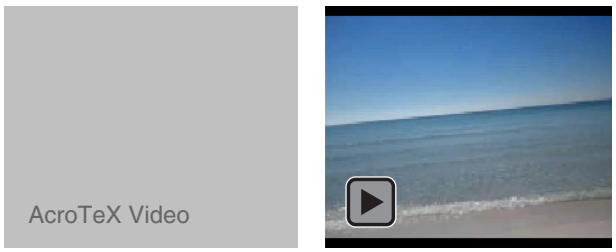
```
\resetWindowDimPos
```

4.2. 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 3GP file. The one on the left uses the default poster, the one on the right uses a custom poster. In the annotation

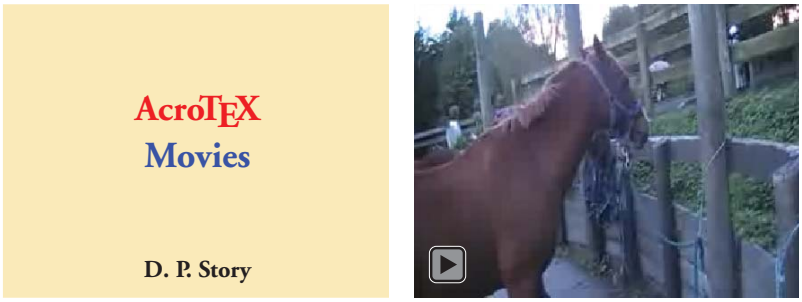
on the left, you see the default posternote, this can be changed using the posternote key.

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

```
\begin{center}
  \rmAnnot[width=1.5in]{176bp}{144bp}{oceanwaves}\quad
  \rmAnnot[poster=ow-poster,width=1.5in,windowed]{176bp}{144bp}{oceanwaves}
\end{center}
```

The poster ow-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 18, for details on how this was done.)



The verbatim listing for the two above annotations follows:

```
\rmAnnot[poster=aebmovie_poster,width=2in]{640bp}{480bp}{horse1}\quad
\rmAnnot[poster=horse1_poster,width=2in]{640bp}{480bp}{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.

Default MP3 poster. 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:

```
\rmAnnot[height=14bp]{\cntrlbrWd}{\cntrlbrHt}{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 media 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.

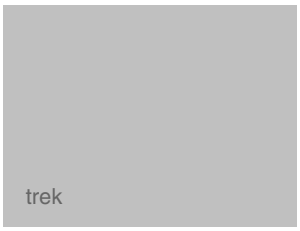
• Additional remarks on MP3

For versions of AA/AR prior to the EOL for Flash content, a control bar was used to control the playback of an MP3. The poster version is still available, as seen below:

❶ 

```
\rmAnnot[height=14bp]{\cntrlbrWd}{\cntrlbrHt}{trek}
```

however, the control bar *does not appear* when the MP3 is played, rather, the new player appears. The rmannot package now offers alternatives to the old control bar:



❷ Default Poster

❸ Customized Default Poster

❹ Custom Poster

The verbatim listing of the the above three **RMA**s:

```
❷ \rmAnnot[width=1.5in,defaultposter,posternote=trek]{640bp}{480bp}{trek}
❸ \setPosterProps{textSize=14,xPos=20,yPos=20,textColor=1 0 0 setrgbcolor}%
❸ \rmAnnot[width=1.5in,defaultposter,posternote=trek]{640bp}{480bp}{trek}
❹ \rmAnnot[width=1.5in,poster=aebmusic_poster]{640bp}{480bp}{trek}
```

Comments on MP3 posters

❶ When no poster option is taken, as in display ❶ above, the old control bar is used as the poster.

❷ When `defaultposter` key appears, the control bar poster *is not used*; what is used in the default poster image, refer to the paragraph titled **Default Poster Image** on page 11 for information on this poster. In this example, `posternote` is used to add text to the poster. Without the `posternote` the default text appears as 'AcroTeX MP3' appears.

❸ The **RMA** labeled as ❸ uses the \setPosterProps command, as described in paragraph **The \setPosterProps command** on page 11, to modify the size, positioning and color of the poster note.

❹ In this **RMA**, the poster key is specified.

4.3. \rmAnnot and 3D

Here is something that I've only just come to realize: If you use the user interface (UI) of Acrobat and you create a 3D annotation in Acrobat, then give it a SWF as a resource, the 3D annot gets converted into a Rich Media annotation (RMA). Looking through the specification as described in the *Adobe Supplement to ISO 32000*, I determined to implement this feature, and why not since most of the structure (that of an RMA) was already in place by way of my rmannot package. So, this version of rmannot supports what I'll call *Rich Media 3D annotation* (**RMA3D**).

Initially, it was not a challenge to get a 3D model to appear in a **RMA** created by rmannot, some straight forward modifications to rmannot were required with *ISO 32000* as a guide. Looking at Alexander Grahn's very fine and brilliant movie15 package, I saw the difficulties of defining and creating *view*s through the LaTeX interface. With Alexander's permission, I gently lifted all the really heavy code from movie15, and placed it in rmannot. I offer up my great and humble thanks for his kindness in allowing the use of his code (characterized by commands beginning with @MXV in rmannot.dtx).

If you want to insert an **RMA3D** annotation into your document, begin by calling the rmannot package with the use3D option

```
\usepackage[use3D]{rmannot}
```

Using this option brings in a large amount of code to support 3D. Regular **RMA**s can be created as usual, if you do not use 3D there is no reason to use this option.

The 3D Models support by Acrobat/Adobe Reader are U3D and PRC. To construct a **RMA3D**, you use one of these filetypes as the fourth argument of \rmannot, for example,

```
\rmAnnot[⟨rmannot_opts⟩]{⟨width⟩}{⟨height⟩}{⟨3dmodel⟩}
```

\rmAnnot files and resources are referred to symbolically, and need to be declared in the preamble. For example, we might declare

```
\saveNamedPath{myDice}{c:/.../3dmodels/dice.u3d}
```

\rmAnnot parses the fourth argument, and looks at its extension. If the extension is .u3d or .prc, the appropriate 3D structure is generated for this annotation.

The first optional argument of \rmAnnot has two new key-value pairs, both Boolean: toolbar and modeltree.

- toolbar: A Boolean, which if true (the default), causes the 3D toolbar to appear when the annot is activated. If toolbar=false, the toolbar does not appear when the annotation is activated.

- modeltree: A Boolean, which if true causes the Model Tree as viewed in the Navigation Pane. The default is false, the Model Tree is not displayed when the annotation is activated.

There are a large number of key-values that support **RMA3D** annotations, rather than inserting them into the first optional parameter of \rmAnnot, I've created a separate command, \setRmOptions3D for this purpose. The command may appear appear anywhere before the **RMA3D** it is referencing. The syntax is,

```
1  \setRmOptions3D{⟨annot_name⟩}
2  {
3      3DOptions={⟨options from movie15⟩},
4      3DResources={%
5      none={rName=⟨name1⟩},...,
6        foreground={rName=⟨name2⟩,flashvars=⟨vars⟩},...,
7        background={rName=⟨name3⟩,flashvars=⟨vars⟩},...,
8        material={rName=⟨name4⟩,mName=⟨materialName⟩,flashvars=⟨vars⟩},...
9     }
10 }
```

The command takes two arguments, the first ⟨*annot_name*⟩ is the name of the annot, as declared by the name key in the first optional argument of \rmAnnot, like so,

```
\rmAnnot[name=my3DDice,...]{4in}{3in}{myDice}
```

In the above example, we've named this annot my3DDice, and it is this name we would put in as the first argument of \setRmOptions3D in line (1) above.

The second argument of \setRmOptions3D takes key-value pairs, but there ~~are only two keys~~ is only one key: 3DOptions ~~and 3DResources~~. Each of these will be explained in turn.

3DOptions As noted in line (3), the value of this key are key-value pairs defined in movie15, appropriate to 3D models. The keys supported are 3Dbg, 3Djscript, 3Dcoo, 3Dc2c, 3Droo, 3Daac, 3Droll, 3Dviews, 3Dlights, and 3Drender. See the movie15 documentation for a description of these keys.

There are a couple of differences. First 3Dviews is the 3Dviews2 of movie15. Alexander Grahn had deprecated his original 3Dviews key, and later came up with a better format for storing the views. Since we are beginning anew, 3Dviews uses the new format as described in the movie15 documentation as 3Dviews2.

Another difference is with the 3Djscript key. The file descriptor must be a symbolic name, defined by \saveNamedPath command. The value of 3Djscript can be a comma delimited list of JavaScript files, for example,

```
3DOptions={%
  ...,
  3Djscript={myScript,myTurntable},
  ...,
  ...
}
```

Again myScript and myTurntable are defined by the \saveNamedPath command. In theory, one can build a library of general and specific JavaScripts to do 3D work, and you can concatenate them together in this way.

The 3Dviews key takes as its argument a views file. This is purely a LaTeX object (not used or required by Distiller), to the usual filename is needed, for example,

```
3DOptions={%
  ...,
  3Djscript={myScript,myTurntable},
  3Dviews=dice.vws,
  ...
}
```

After Dec. 2020, the 3DResources key is deprecated

3DResources. ~~This is a key that is new, and separate from the movie15 keys just outlined. 3DResources recognizes four keys, these are none, foreground, background, and material. The names and values found within 3DResources are modeled after the **Resources** tab of the **Edit 3D** dialog box of Acrobat 9 or later.~~

```
\setRmOptions3D{⟨annot_name⟩}
{
    3DOptions={⟨options from movie15⟩},
    3DResources={%
    none={rName=⟨name1⟩},...,
      foreground={rName=⟨name2⟩,flashvars=⟨vars⟩},...,
      background={rName=⟨name3⟩,flashvars=⟨vars⟩},...,
      material={rName=⟨name4⟩,mName=⟨materialName⟩,flashvars=⟨vars⟩},...
   }
}
```

~~A resource is usually a SWF file, but can be a FLV, or another 3D model (.u3d, .prc); rmannot does not support image files are resources (JPG, PNG, etc).~~

Note: ~~Convert all image files (JPG, PNG, etc) to a SWF for used by rmannot. The conversion can be made by Adobe Flash Professional, or by using **SWF Tools** (use the jpeg2swf and png2swf tools).~~

~~SWF files may be bound to the background, foreground, a material of the 3D model, or not bound at all. FLV and 3D models must be not bound, and listed under the none key.~~

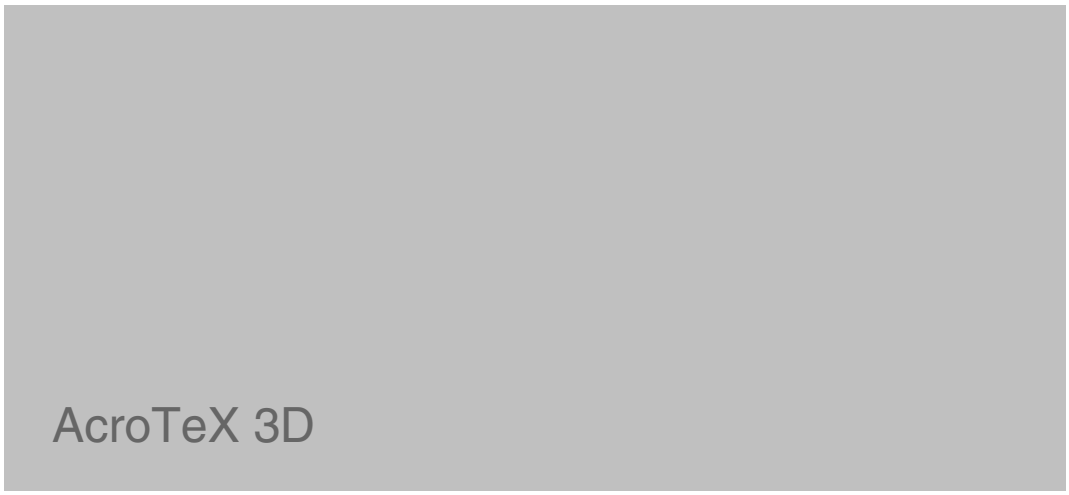
~~The keys none, foreground, background, and material may appear multiple times. A brief description of the values of each key follows:~~

- ~~none: The value of none is a single key-value combination. rName=⟨name⟩, where ⟨name⟩ is the symbolic name of a resource file declared by the \saveNamedPath. These files can be SWF, FLV, or even another model (advanced).~~
- ~~foreground: This key binds a resource to the foreground of the 3D scene. The foreground key takes at most two key-value pairs, only rName is required, the symbolic name of the resource. The flashvars key is used to pass flash variables to the SWF application.~~

- ~~background: This key binds a resource to the background canvas of the 3D scene. The background key takes at most two key-value pairs, only rName is required, the symbolic name of the resource. The flashvars key is used to pass flash variables to the SWF application.~~
- ~~material: This key binds a resource to a material. The resource name is rName (as defined by \saveNamedPath), the key mName is the name of the material the resource is to be bound to; flashvars is used to pass variables to the SWF application.~~

~~If a SWF resource is to be used as background, foreground, or a material using 3D JavaScript (through the JS file input by the 3Djscript key), it must be listed through the none key.~~

Example. We finish off this section with a simple example,



The verbatim listing is,

```
\setRmOptions3D{my3DDice}{%
  3DOptions={%
    3Droo=27,
    3Dlights=Cube,
    3Drender=Solid,
    3Dbg=1 0 0,
    3Dviews=../examples/rm3da/views/dice.vws,
  }
}
\rmAnnot[name=my3DDice,toolbar]{\linewidth}{2.5in}{myDice}
```

Further examples will appear, in time, on my AcroTEX Blog.

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