AcroTEX.Net

The AeB Pro Package

Linking to Attachments

D. P. Story

Prepared: January 13, 2007

http://www.acrotex.net
Published: January 12, 2007

Table of Contents

1. Introduction

As we saw briefly in aebpro_ex3.tex, it is possible to attach a document using the docassembly environment, as illustrated below,

```
\begin{docassembly}
\importDataObject({
    cName: "\aref(cooltarget)",
    cDIPath: "aebpro_ex2.pdf"
});
\end{docassembly}
```

In the above, we use \importDataObject, set the path to be cDIPath: "aebpro_ex2.pdf" (this can be absolute or relative), and give the attachment a name with cName: "\aref(cooltarget)". The special command \aref, is used to reference the assigned name has as its argument the label name, *delimited by parentheses*.

The parameter cName in the above docassembly code is of particular importance. The value of cName is used in the names tree for embedded files. It is used to reference the attachment in the link code. After the file is imported, the value of cName is converted by Acrobat to Unicode. When referencing it, you must know the unicode of the value of the cName key.

First, we insert into the preamble, the linktoattachments option. This

brings in all the code and commands to be discussed in this document. (See the preamble of this file.)

2. Naming Attachments

For documents attached by the `attachments` option, AeB Pro assigns them "names," which appear in the attachments tab of Acrobat/Reader as the Description.[1] The names assigned are `AeB Attachment 1`, `AeB Attachment 2`, `AeB Attachment 3`, and so on. If you embedded the file using the `docassembly` environment and the `\importDataObject` method, then you are free to assign a name of your preference. However it is done, the names must be converted to unicode on the TEX side of things to set up the links, and there must be a LATEX-like way of referencing this unicode name, hence the development of the `attachmentNames` environment and the two commands `\autolabelNum` and `\labelName`.[2]

We describe these by example. In the preamble you will find

```
\autolabelNum{1}
\autolabelNum*{2}{target2.pdf Attachment File}
```

[1] The Description is important as it is the way embedded files are referenced internally.

[2] It is important to note that these are not needed unless you are linking to the embedded (PDF) files.

```
\autolabelNum*[AeST]{3}{AeBST Components}
\labelName{cooltarget}{My (cool) $|x^3|$ ~ % '<attachment>'}
\end{attachmentNames}
```

Note: The `attachmentNames` environment and the commands `\autolabelNum` and `\labelName` should be used only in the parent document; for child document, they are not necessary.

`\autolabelNum`: For PDFs (or other files) embedded using the `attachments` option, use the `\autolabelNum` command. The syntax is

```
\autolabelNum[<label>]{<num>}
```

The first optional argument is the label to be used to refer to this embedded file; the default is `attach<num>`. The second argument is the second is a number, 1, 2, 3.., which corresponds to the order the file is listed in the value of the `attachments` key.[3]

`\autolabelNum*`: There is a star form of `\autolabelNum`, which allows to to change the description of the attachment.

```
\autolabelNum*[<label>]{<num>}{<description>}
```

[3] To minimize the number of changes to the document, if you later add an attachment, add it to the end of the list so the earlier declarations are still valid.

By default, the first attachment has label name attach<num> and has a description of AeB Attachment <num>, This command allows you not only to change the label, but to change the description of the attachment as well.

\labelName: For files that are embedded in using \importDataObject, use the command \labelName for assigning the name, and setting up the correspondence between the name and the label.

\labelName{<label>}{<description>}

The first argument is the label to be used to reference this embedded file. The second parameter you can assign an arbitrary name.

The <description> parameter used in \autolabelNum* and \labelName can be an arbitrary string assigned to the description of this embedded file, the characters can be most anything in the Basic Latin unicode set, 0021–007E, with the exception of left and right braces {}, backslash \ and double quotes ". If you take the latin1 option, the unicodes for 00A1–00FF are also included.

You can also enter the unicode character codes directly by typing \uXXXX in the <description>, where XXXX are four hex digits. (Did I say not to use '\'?) This is very useful when using the trouble making characters such as backslash, left and right braces, and double quotes, or using unicode above

00FF (Basic Latin + Latin-1). To illustrate, suppose we wish the description of `cooltarget` to be

```
"$|e^{\ln(17)}|$"
```

All the bad characters!

```
\labelName{cooltarget}{\u0022$|e^\u007B\u005Cln(17)\u007D|$\u0022}
```

From the unicode character tables we see that

- left brace `\u007B`
- right brace `\u007D`
- backslash `\u005C`
- double quotes `\u0022`

See the example file `aebpro_ex6.tex` for additional examples of the use of `\uXXXX` character codes.

There are several "helper" commands as well: `\EURO`, `\DQUOTE`, `\BSLASH`, `\LBRACE` and `\RBRACE`. When the `\u` is detected, an `\expandafter` is executed. This allows a command to appear immediately after the `\u`, things work out well if the command expands to four hex numbers, as it should. Thus, instead of typing `\u0022` you can type `\u\DQUOTE`.

3. Linking to Embedded Files

This package defines two commands, `\ahyperref` and `\ahyperlink`, to create links between parent and child and child and child. The default behavior of `\ahyperref` (and `\ahyperlink`) is to set up a link from parent to child. `\ahyperlink` and `\ahyperref` are identical in all respects except for how it interprets the destination. (Refer to 'Jumping to a target' on page 9 for details.)

The commands each take three arguments, the first one of which is optional

```
\ahyperref[<options>]{<target_label>}{<text>}
\ahyperlink[<options>]{<target_label>}{<text>}
```

In the simplest case, we jump from the parent to the first page of a child file, like so target1.pdf, given by...

```
\ahyperref{attach1}{target1.pdf}
```

This is the same as target1.pdf, the code is...

```
\ahyperref[goto=p2c]{attach1}{target1.pdf}
```

The `goto` key is one of the key-value pairs taken by the optional argument. Permissible values for the `goto` key are `p2c` (the default), `c2p` (child to parent) and `c2c` (child to child).

TIP: After jumping to an attachment you can return to the point of origin (in the parent document) by selecting the menu item `View > GoTo > Previous Document` or by using the keyboard shortcut of `Shift+Alt+Left Arrow`

Similarly, link to the other embedded files in this parent: target2.pdf and aebpro_ex2.pdf

In all cases above, the `\ahyperlink` command could have been used, because no *named* destination was specified, without a named destination, the these links jump to page 1.

3.1. Jumping to a target

As you may know, LaTeX, more exactly, `hyperref` has two methods of jumping to a target in another document, jump to a label (defined by `\label`) and jump to a target set by `\hypertarget`. Each of these is demonstrated for embedded files in the next two sections.

• Jumping to a `\hypertarget` with `\ahyperlink`

There is a destination defined by the hyperref command `hypertarget` in `target1.pdf` and we shall jump to it. Here we go! Jump!. The code for this jump is

```
\ahyperlink[dest=mytarget]{attach1}{Jump!}
```

Note that `dest=mytarget`, where "`mytarget`" is the label assigned by the `\hypertarget` command in `target1.pdf`.

- **Jumping to a `\label` with `\ahyperref`**

LaTeX has a cross-referencing system, to jump to a target set by the `\label` command we use the xr-hyper package that comes with `hyperref`. Using label referencing, we can jump to Section 1 on page 2 of the embedded file `target1.pdf`. Swave! The code for the jump is

```
\ahyperref[dest=target1-s:intro]{attach1}
    {Section~\ref*{target1-s:intro}}
```

we set `dest=target1-s:intro`

The label in `target1.pdf` is `s:intro`, in the preamble of this document we have

```
\externaldocument[target1-]{children/target1}
```

which causes xr-hyper to append a prefix to the label (this avoids the possibility of duplication of labels, over multiple embedded files).

3.2. Optional Args of \ahyperref and \ahyperlink

The \ahyperref commands has a large number of optional arguments enabling you to create any link that the user interface of Acrobat Pro can create, and more. These are documented in aeb_pro.dtx and well as the main documentation. Suffice it to have an example or two.

By using the optional parameters, you can create any link the UI can create: Jump! This link is given by...

```
\ahyperref[%
    dest=target1-s:intro,
    bordercolor={0 1 0},
    highlight=outline,
    border=visible,
    linestyle=dashed
]{attach1}{Jump!}
```

Now what do you think of that?

The argument list can be quite long, as shown above. If you have some favorite settings, you can save them in a macro, like so,

```
\def\preseti{bordercolor={0 0 1},highlight=outline,open=new,%
    border=visible,linestyle=dashed}
```

Then, I can say, more simply, Jump! This link is given by...

```
\ahyperref[dest=target1-s:intro,preset=\preseti]{attach1}{Jump!}
```

I've defined a `preset` key so you can predefine some common settings you like to use, the enter these settings through preset key, like so `preset=\preseti`. Cool.

Note that in the second example, I've included `open=new`, this causes the embedded file to open in a new window. For Acrobat/Reader operating in MDI, a new child window will open; for SDI (version 8), and if the user preferences allows it, it will open in its Acrobat/Adobe Reader window.

TIP: After jumping to an attachment that opens a new window, just close the new window to return the parent document. :-{)

4. Opening and Saving with `\ahyperextract`

In addition to embedding and linking a PDF, you can embed most any file and programmatically (or through the UI) open and/or save it to the local file system.

To attach a file to the parent PDF, you can use the `attachsource` or the `attachments` options of AeB Pro, or you can embed your own using the `importDataObject` method, as described earlier in this file.

If an embedded file is a PDF, then you can link to it, using \ahyperref or \ahyperlink; we can jump to an embedded PDF and jump back. If the embedded file is not a PDF, then jumping to it makes no sense; the best we can do is to open the file (using an application to display the file) and/or save it to the local hard drive.

The AeB Pro package has the command \ahyperextract to extract the embedded file, and to save it to the local hard drive, with an option to start the associated application and to display the file. The syntax for \ahyperextract is the same as that of the other two commands:

\ahyperextract[<options>]{<target_label>}{<text>}

The <options> are the same as \ahyperref, the <target_label> is the one associated with the attachment name, and the <text> is the link text.

In addition to the standard options of \ahyperref, \ahyperextract recognizes one additional key, launch. This key accepts three values: save (the default), view and viewnosave. The following is a partial verbatim listing of the descriptions given in the *JavaScript for Acrobat API Reference*, in the section describing importDataObject method of the Doc object:

- save: The file will not be launched after it is saved. The user is prompted for a save path.

- `view`: The file will be saved and then launched. Launching will prompt the user with a security alert warning if the file is not a PDF file. The user will be prompted for a save path.
- `viewnosave`: The file will be saved and then launched. Launching will prompt the user with a security alert warning if the file is not a PDF file. A temporary path is used, and the user will not be prompted for a save path. The temporary file that is created will be deleted by Acrobat upon application shutdown.

Here is a series of examples of usage:

1. aebpro_ex3.pdf: Launch and view this PDF. The code is

   ```
   \ahyperextract[launch=view]{cooltarget}{aebpro\_ex3.pdf}
   ```

   When you extract (or open) PDF in this way, any links created by `\ahyperref` or `\ahyperlink` as the PDF being viewed is no longer an embedded file of the parent.

2. View the aebpro_ex5.tex. The code is

   ```
   \ahyperextract[launch=viewnosave]{tex}{aebpro\_ex5.tex}
   ```

   Note that for attachments brought in by the `attachsource` option, the label for that attachment is the file extension, in this case `tex`.

3. AeBST Component List: This is an Excel spreadsheet which lists the components of the AcroTEX eDucation System Tools. Here you are prompted to save; the spreadsheet will not be launched:

   ```
   \ahyperextract[launch=save]{AeST}{AeBST Component List}
   ```

5. Final Assembly

To assemble your parent document with all the cross-references to its embedded children, perform the following steps.

1. LaTeX the parent document so that the auxiliary file `\jobname_xref.cut`. This file is read by the children when they are LaTeXed.
2. LaTeX the child documents. The child documents will write their own auxiliary file and read `\jobname_xref.cut`. (Multiple compiles may be necessary to bring the auxiliary document up to date.)
3. Make PDF for the child documents.
4. Now LaTeX the parent again, which will read in the auxiliary files of the children, if needed. Distill and *Le Voilà*, you have it!
5. At this point you can clean up all auxiliary files.