

The docassembly package

D. P. Story
Email: dpstory@acrotex.net

processed June 16, 2021

Contents

1	Introduction	1
2	Documentation	2
3	Package requirements	3
4	Document Assembly Methods	3
4.1	The docassembly environment	3
4.2	Supported Assembly JS API	3
4.2.1	\addWatermarkFromFile	5
4.2.2	\addWatermarkFromText	5
4.2.3	\importIcon	5
4.2.4	\importSound	5
4.2.5	\appopenDoc	5
4.2.6	\docSaveAs	5
4.2.7	\insertPages	6
4.2.8	\extractPages	6
4.2.9	\createTemplate	6
4.2.10	\importDataObject	6
4.2.11	\executeSave	6
4.2.12	\mailDoc	6
4.2.13	Signature related commands	7
5	Index	10
6	Change History	10
1	⟨*package⟩	

1 Introduction

The `docassembly` package provides access some security related features of the Acrobat JavaScript API. The content of this package was extracted from `aeb_pro` so these important and useful methods can be available to document authors that own Acrobat, but enjoy the use of `pdflatex`, `xelatex`, `lualatex`, `ps2pdf`, or (in the worst case scenario) Adobe Distiller as PDF creators.

Important: Run for your lives! This package requires Adobe Acrobat! Run! Do not look back!. However, if it makes you feel better, you can use any of the usual PDF creators (`pdflatex`, `lualatex`, `xelatex`, `dvips` -> `ps2pdf`, or even the much hated `dvips` -> Adobe Distiller).

This package requires `insdljs`, which loads `hyperref`. It also requires the successful installation of the file `aeb_pro.js`, which is distributed with this package, as well as with the `aeb_pro` package. Version 1.6.1 of `aeb_pro.js` is required.¹ Installation of `aeb_pro.js` is covered in `docs/install_jsfiles.pdf`.

2 Documentation

The documentation of this file is minimal. Most of the “teaching” of how to use the methods supported by this package is contained in the demo files.

The package defines one environment, the `docassembly` environment, and several “helper” commands that are used within the `docassembly` environment. Refer to [Section 4](#) for details.

Installation of `aeb_pro.js`. This file is essential to the correct functioning of this package. It is important, therefore, to correctly install it. Follow the instructions of `docs/install_jsfiles.pdf`.

Demo files. Demo files are contained in the `examples` folder. They illustrate all the security restricted methods of this package. Most of the demo files have a minimal preamble. Several of the files also uses `eforms`, since form fields are used in those demo files.

End this package early. If `aeb_pro` is already loaded, terminate this package early since `aeb_pro` contains the content of this package already.

```
2 \@ifpackageloaded{aeb_pro}{\PackageInfo{docassembly}
3   {aeb_pro detected, early exit from\MessageBreak
4     the docassembly package}\endinput}{\}
5 \edef\da@restoreCats{%
6   \catcode'\noexpand\"=\the\catcode'\\"relax
7   \catcode'\noexpand\,=\the\catcode'\,\relax
8   \catcode'\noexpand\(\=\the\catcode'\(\relax
```

¹If you already have `aeb_pro` installed on your system, be sure you have Version 1.6.1 of `aeb_pro.js`, if not, use the version of `aeb_pro.js` that comes with `docassembly`.

```

9      \catcode'\noexpand\!=\the\catcode'\!\relax
10 }
11 \@makeother"\@makeother\,\@makeother\(\@makeother\!

```

3 Package requirements

Require `insdljs` for this little exercise.

```

12 \RequirePackage{insdljs}
13 \execJSOn

```

4 Document Assembly Methods

Special “helper” commands and one environment are defined to take advantage of Acrobat’s extensive library of security restricted methods. It is assumed the document author has properly installed `aeb_pro.js`.

4.1 The docassembly environment

`docassembly` This is a wrapper environment for the `execJS` environment of the `insDLJS` Package. Place JavaScript lines in this environment and the script will execute one time after the PDF has been created and opened in Acrobat Pro for the first time.

```

14 \newenvironment{docassembly}{\execJS{docassembly}}{\endexecJS}

```

4.2 Supported Assembly JS API

These are convenience commands – called JavaScript helper commands – to executing security restricted JavaScript. The JS methods are defined in the `aeb_pro.js` file, kept as folder JavaScript. These commands are executed in a verbatim environment where ‘\’ is still the escape character. Each of the JavaScript helper commands expects a left parenthesis ‘(’ following the command name *on the same line* as the command name.² See the example below for correct usage.

```

\begin{docassembly}
\addWatermarkFromFile({
  bOnTop:false,
  cDIPath:"/C/AcroTeX/AcroPackages/ManualBGs/Manual_BG_Print_AeB.pdf"
});
\end{docassembly}

```

For each of the methods below, see the *JavaScript for Acrobat API Reference*.

The command `\theDocObject` is normally set to `this`, meaning the current document. You may need to set it to some other doc object if you are trying to access a doc object other than the current one. The following are support commands for changing `\theDocObject` from within the `docassembly` environment.

²This requirement is consistent with JavaScript function usage.

`\chngDocObjectTo` All the JavaScript helper commands use `\theDocObject`, which is defined to be the `this` object. To change it within the `docassembly` environment is difficult. The next command aids in that problem.

```
15 \let\ap@mrk\@empty
16 \def\ap@gobtocomma#1,{
17 \providecommand\chngDocObjectTo[2]{%
18   \def#1##1\ap@mrk{#2,\ap@gobtocomma##1}}
```

The above defines a new command given by `#1`. The command has one argument which is all content up to the terminating mark `\ap@mrk`. The trick to removing `\thisDocObject` and replacing it with `#2`, in the above definition, we insert `'(#2'` followed by `\ap@gobtocomma`, which absorbs `\thisDocObject`, (absorbs everything through the first comma), followed by all content (`##1`); the second `\@gobble` absorbs the left parenthesis that opens the argument.

```
19 \def\ap@TF{aebTrustedFunctions}
```

An example of usage of `\chngDocObject` is `\chngDocObjectTo{\newD0}{doc}` expanded above the `docassembly` environment. Later, we can say,

```
\chngDocObjectTo{\newD0}{doc}
\begin{docassembly}
...
\docSaveAs\newD0({ cPath: _path });
...
\end{docassembly}
```

That is, it is placed immediately after any of the commands below that uses `\theDocObject`.

`\theDocObject` This command is used in the definition of all JavaScript helper commands, as seen in the definition of `\DeclareJSHelper` below. It is set to the doc object `this`. It can be changed using `\chngDocObjectTo`, as described above.

```
20 \def\theDocObject{this}
```

`\DeclareJSHelper` A general purpose command for defining what I am calling JavaScript helper commands.

```
21 \providecommand\DeclareJSHelper[2]{%
22   \def#1##1({\ap@TF{##1\theDocObject,#2,\ap@mrk}}}
```

For example, we declare `\DeclareJSHelper{\docSaveAs}{aebDocSaveAs}` below, the declaration defines a new command, `\docSaveAs`:

```
\def\docSaveAs#1({\ap@TF{#1\theDocObject,aebDocSaveAs,\ap@mrk}}
```

Note that the argument of `\docSaveAs` is delimited by the left parenthesis, thus `#1` is everything through that opening parenthesis. This approach allows more flexibility in the definition, there can be spaces following the command name `\docSaveAs ({path: _path})`, for example.

`\retnAbsPathAs` Several methods require an absolute path to the current folder. The code is,

```
var _path=this.path;
var pos=_path.lastIndexOf("/");
_path=_path.substring(0,pos);
```

We simplify this code for the document author in the form of the command `\retnAbsPathAs(<js-var>);`, where `<js-var>` is a JavaScript variable that will hold the absolute path to the current folder; eg, `\retnAbsPathAs(_path);` expands to the above code.

```
23 \def\retnAbsPathAs(#1){var #1=this.path;^^J%
24   var pos=#1.lastIndexOf("/");^^J%
25   #1=#1.substring(0,pos)}
```

We now begin the documentation of the “helper” commands. For documentation of the arguments of these commands, refer to the *JavaScript™ for Acrobat® API Reference*.

4.2.1 `\addWatermarkFromFile`

`\addWatermarkFromFile` This is the method `Doc.addWatermarkFromFile`.

Demo file: watermark-file.tex

```
26 \DeclareJSHelper{\addWatermarkFromFile}{aebAddWatermarkFromFile}
```

4.2.2 `\addWatermarkFromText`

`\addWatermarkFromText` This is the method `Doc.addWatermarkFromText`.

Demo file: watermark-text.tex

```
27 \DeclareJSHelper{\addWatermarkFromText}{aebAddWatermarkFromText}
```

4.2.3 `\importIcon`

`\importIcon` This is the method `Doc.importIcon`.

Demo file: import-icons.tex

```
28 \DeclareJSHelper{\importIcon}{aebImportIcon}
```

4.2.4 `\importSound`

`\importSound` This is the method `Doc.importSound`.

Demo file: import-sound.tex

```
29 \DeclareJSHelper{\importSound}{aebImportSound}
```

4.2.5 `\appopenDoc`

`\appopenDoc` This is the method `app.openDoc`. Opens a document and return a Doc object.

Demo file: open-doc.tex

```
30 \DeclareJSHelper{\appopenDoc}{aebAppOpenDoc}
```

4.2.6 \docSaveAs

`\docSaveAs` This is the method `Doc.saveAs`. Saves a PDF or converts a PDF to another format to a specified path.

Demo file: doc-saveas.tex

```
31 \DeclareJSHelper{\docSaveAs}{aebDocSaveAs}
```

4.2.7 \insertPages

`\insertPages` This is the method `Doc.insertPages`.

Demo file: insert-pages.tex

```
32 \DeclareJSHelper{\insertPages}{aebInsertPages}
```

4.2.8 \extractPages

`\extractPages` This is the method `Doc.extractPages`.

Demo file: extract-pages.tex

```
33 \DeclareJSHelper{\extractPages}{aebExtractPages}
```

4.2.9 \createTemplate

`\createTemplate` This is the method `Doc.createTemplate`. This is a feature that I had great hopes for. With templates, you can create hidden pages that can be made visible (in AA); once a template is created, it can be spawned and deleted in AR.

```
34 \DeclareJSHelper{\createTemplate}{aebCreateTemplate}
```

4.2.10 \importDataObject

`\importDataObject` This is the method `Doc.importDataObject`, used to attach files to a PDF; alternately, `\attachFile` is a more intuitive name for the operation performed.

Demo file: attach-files.tex

```
35 \DeclareJSHelper{\importDataObject}{aebImportDataObject}
```

```
36 \DeclareJSHelper{\attachFile}{aebImportDataObject}
```

4.2.11 \executeSave

`\executeSave` To save the document, use at the *end of the doc assembly environment*. Usage: `\executeSave()`. The `\@gobble` used below absorbs the comma that is placed immediately after the second argument by `\DeclareJSHelper`.

```
37 \DeclareJSHelper{\executeSave}{aebSaveAs,"Save"\@gobble}
```

4.2.12 \mailDoc

`\mailDoc` This is the method `Doc.mailDoc`. Attach the current PDF and email it someone. Must have a default mail client registered by Acrobat under Edit > Preferences > Mail Accounts.

Demo file: mail-doc.tex

```
38 \DeclareJSHelper{\mailDoc}{aebMailDoc}
```

4.2.13 Signature related commands

Demo files: `sign.tex`, `certifyinvisible.tex`

The `\sigInfo` command is used for entering signing formation into what will become an object. `\signatureSign` takes no arguments, but uses the info entered by `\sigInfo`. An example is

```
\begin{docassembly}
\sigInfo{
  cSigFieldName: "sigOfDPS",
  ohandler: security.PPKLiteHandler,
  cert: "<name>.pfx", password: "<password>",
  oInfo: { location: "Niceville, FL",
    reason: "I am approving this document",
    contactInfo: "dpstory@acrotex.net",
    appearance: "My Signature" }
};
\signatureSign
\end{docassembly}
```

`\sigInfo` The `\sigInfo` command is a latex interface to creating the `oSigInfo` object.

```
39 \newcommand{\sigInfo}{var oSigInfo=}
40 \def\sigFieldObj(#1){var oSigField=this.getField(#1)}
```

For the `\signatureSetSeedValue`, the field object is required. This function assumes that the JavaScript variable `oSigField` is the field object. For example,

```
\begin{docassembly}
\sigFieldObj("sigOfDPS");
\signatureSetSeedValue({
  lockDocument:true,
  appearanceFilter:"My Signature",
  reasons: ["This is a reason", "This is a better reason"],
  flags:0x80|0x100|0x8
});
\end{docassembly}
```

The `signatureSetSeedValue()` method seeds a signature field with various default values available to the signer.

```
\begin{docassembly}
var sv={
  mdp: "defaultAndComments",
  reasons: ["This is a reason", "This is a better reason"],
  flags:0x80|0x100|0x8
};
\sigFieldObj("sigOfDPS");
\signatureSetSeedValue(sv);
\end{docassembly}
```

`\signatureSetSeedValue` This is the *Field.signatureSetSeedValue* method. The field name is passed to this method through the *cSigFieldName* property of the *oSigField* object.

```
41 \def\signatureSetSeedValue#1{%
42     \ap@TF( oSigField, aebSignatureSetSeedValue, }
```

`\signatureSign` This is the *Field.signatureSign* method. The field name is passed to this method through the *cSigFieldName* property of the *oSigField* object. The function `\signatureSign` takes the info in the *oSigInfo* object, gets the security handler object, logs into the handler, calls *signatureSetSeedValue* if the *sv* property is in the *oSigInfo* object, and signs the field.

```
43 \begin{defineJS}[\makecmt\%\dfnJSCR{^^J}]{\signatureSign}
44 if ( typeof oSigInfo.oHandler=="undefined" )
45     oSigInfo.oHandler=security.PPKLiteHandler;
46 var engine=aebTrustedFunctions( security,%
47 aebSecurityGetHandler, oSigInfo.oHandler );
48 var path2Cert = (typeof oSigInfo.path2Cert == "undefined") ? %
49 aebTrustedFunctions( this, aebAppGetPath,%
50 {cCategory:"user"} )+"/Security+"/"+oSigInfo.cert : %
51 oSigInfo.path2Cert;
52 aebTrustedFunctions( engine, aebSecurityHandlerLogin,%
53 { cPassword: oSigInfo.password, cDIPath: path2Cert});
54 var oSigField = this.getField(oSigInfo.cSigFieldName);
55 oSigInfo.oInfo.password=oSigInfo.password;
56 if ( typeof oSigInfo.sv!="undefined" ) {
57     for (var o in oSigInfo.sv )
58         oSigInfo.oInfo[o]=oSigInfo.sv[o];
59 }
60 var oSigArgs={ oSig: engine, oInfo: oSigInfo.oInfo };
61 if ( typeof oSigInfo.cLegalAttest!="undefined" )
62     oSigArgs.cLegalAttest=oSigInfo.cLegalAttest;
63 if ( typeof oSigInfo.cDIPath!="undefined")
64     oSigArgs.cDIPath=oSigInfo.cDIPath;
65 if ( typeof oSigInfo.bUI!="undefined")
66     oSigArgs.bUI=oSigInfo.bUI;
67 aebTrustedFunctions( oSigField, aebSignatureSign, oSigArgs );
68 \end{defineJS}
```

`\certifyInvisibleSign` This is the *Doc.certifyInvisibleSign* method. This command uses the trusted version of *certifyInvisibleSign* to sign. The command requires that *\sigInfo* is populated appropriately.

```
\begin{docassembly}
\sigInfo{
    cert: "<name>.pfx",
    password: "<password>",
    cLegalAttest: "Certified using JavaScript",
    bUI:false,
    oInfo: {
        location: "Niceville, FL",
```



```

        reason: "I am certifying this document",
        mdp: "defaultAndComments",
    }
};
\certifyInvisibleSign
\end{docassembly}

69 \begin{defineJS}{\makecmt\%\dfnJSCR{^^J}}{\certifyInvisibleSign}
70 if ( typeof oSigInfo.oHandler=="undefined" )
71   oSigInfo.oHandler=security.PPKLiteHandler;
72 var engine=aebTrustedFunctions( security, %
73 aebSecurityGetHandler, oSigInfo.oHandler );
74 var path2Cert=aebTrustedFunctions( this, aebAppGetPath, %
75 {cCategory:"user"} )+"/Security"+"/"+oSigInfo.cert;
76 aebTrustedFunctions( engine, aebSecurityHandlerLogin, %
77 { cPassword: oSigInfo.password, cDIPath: path2Cert});
78 oSigInfo.oInfo.password=oSigInfo.password;
79 var oSigArgs={ oSig: engine, oInfo: oSigInfo.oInfo };
80 if ( typeof oSigInfo.cLegalAttest!="undefined" )
81   oSigArgs.cLegalAttest=oSigInfo.cLegalAttest;
82 if ( typeof oSigInfo.cDIPath!="undefined")
83   oSigArgs.cDIPath=oSigInfo.cDIPath;
84 if ( typeof oSigInfo.bUI!="undefined")
85   oSigArgs.bUI=oSigInfo.bUI;
86 aebTrustedFunctions( this, aebCertifyInvisibleSign, oSigArgs );
87 \end{defineJS}

88 \da@restoreCats
89 \</package>

```

5 Index

Numbers written in *italic* refer to the page where the corresponding entry is described; numbers underlined refer to the code line of the definition; numbers in *roman* refer to the code lines where the entry is used.

Symbols			
\!	9, 11	\execJS	14
\%	43, 69	\execJSOn	13
\@makeother	11	\executeSave	<u>37</u>
		\extractPages	<u>33</u>
A		I	
\addWatermarkFromFile	26	\importDataObject	35
\addWatermarkFromText	<u>27</u>	\importIcon	<u>28</u>
\ap@gobtocomma	16, 18	\importSound	29
\ap@mrk	15, 18, 22	\insertPages	<u>32</u>
\ap@TF	19, 22, 42		
\appopenDoc	<u>30</u>	M	
\attachFile	6, 36	\mailDoc	38
		\makecmt	43, 69
C		P	
\certifyInvisibleSign	69	\PackageInfo	2
\chngDocObjectTo	<u>15</u>	\providecommand	17, 21
\createTemplate	<u>34</u>		
D		R	
\da@restoreCats	5, 88	\RequirePackage	12
\DeclareJSHelper	<u>21</u> , 26–38	\retnAbsPathAs	<u>23</u>
\dfnJSCR	43, 69		
docassembly (environment)	<u>14</u>	S	
\docSaveAs	<u>31</u>	\sigFieldObj	40
		\sigInfo	<u>39</u>
E		\signatureSetSeedValue	<u>41</u>
\endexecJS	14	\signatureSign	<u>43</u>
\endinput	4		
environments:		T	
docassembly	<u>14</u>	\theDocObject	<u>20</u> , 22

6 Change History

v1.0 (2021/06/11)

General: First publication version of this package 2