Digital Signature Appearances

Adobe® Acrobat® SDK

Contents

List of Examples

Preface

This document provides guidelines for specifying how digital signatures are displayed in an Adobe® PDF file.

Who should read this guide?

This document contains information for developers as well as some general information that is relevant to system administrators.

Related documentation

For information about	See
A guide to the documentation in the Adobe Acrobat® SDK.	*Acrobat SDK Documentation Roadmap*
A general overview of the Acrobat SDK.	*Overview*
Detailed descriptions of the APIs for Acrobat and Adobe Reader® plug-ins, as well as for PDF Library applications.	*Acrobat and PDF Library API Reference*

Digital Signature Appearances

This document provides guidelines for the generation of standardized appearances for digital signatures in PDF documents.

Signature interoperability

The digital signature API in Acrobat allows plug-ins to implement *digital signature handlers* to create and verify document signatures.

A goal is to achieve file interoperability between handlers, particularly for those that are implemented based on public-key cryptography. This interoperability allows a PDF file that is signed with one digital signature handler to be verified with a different digital signature handler, assuming the other handler understands the same basic signature format.

To achieve interoperability, a standard PDF syntax for implementation of signatures is required. This standard syntax has two primary components:

- The *signature dictionary*, which is where the actual signature is stored. It includes attributes such as the name of the signer, the time of the signature, the signed hash of the file, and the signer's certificate (for example, embedded as a PKCS#7 object). Standard syntax for the signature dictionary is defined using the SubFilter attribute of the signature dictionary. For more information, see the *PDF Reference*.
- The *signature appearance*, which is how the signature is displayed to the user. The signature appearance is specified in the signature annotation. The display of the signature is optional and standardizing the appearance is not necessary for the cryptographic purpose of verifying a signature, but makes for a consistent user experience.

Annotation appearances

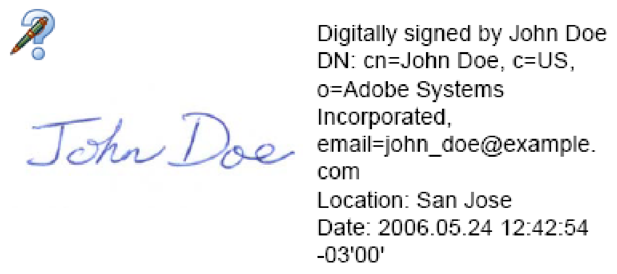
Acrobat digital signatures exist as signature form fields in a PDF file. As with any form field, signature form fields are associated with an annotation dictionary. Appearances for any annotation, including signatures, are contained in the appearance dictionary, or AP attribute, of the annotation. Following the guidelines in this document for signature appearance creation will ensure the best operation within Acrobat.

When developing plug-ins, make use of the complete signature appearance support that is part of the PubSec layer of code that resides between the Acrobat PubSecHFT and the DigSigHFT. For more information, see "Appearance management with PubSec" on page 12.

Example signatures

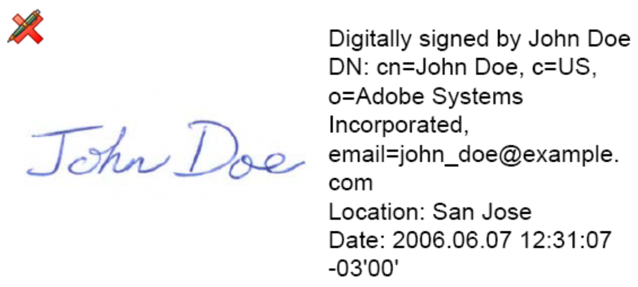
These example signatures are referred to later in this document.

Example 1 ***Unverified signature***


John Doe

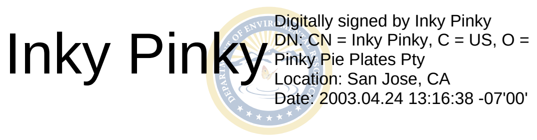
Digitally signed by John Doe
DN: cn=John Doe, c=US,
o=Adobe Systems
Incorporated,
email=john_doe@example.
com
Location: San Jose
Date: 2006.05.24 12:42:54
-03'00'

Example 2 ***Invalid signature***


John Doe

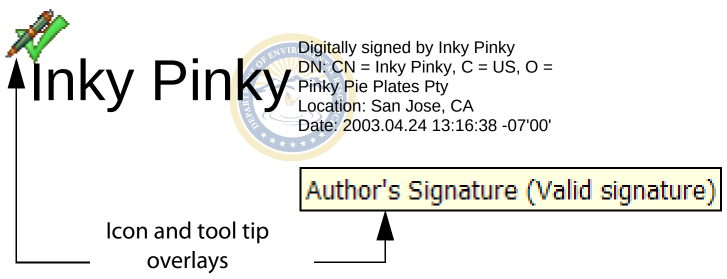
Digitally signed by John Doe
DN: cn=John Doe, c=US,
o=Adobe Systems
Incorporated,
email=john_doe@example.
com
Location: San Jose
Date: 2006.06.07 12:31:07
-03'00'

Example 3 ***Signature appearance with a watermark created using Acrobat PubSec***


Inky Pinky

Digitally signed by Inky Pinky
DN: CN = Inky Pinky, C = US, O =
Pinky Pie Plates Pty
Location: San Jose, CA
Date: 2003.04.24 13:16:38 -07'00'

Example 4 ***Signature appearance created using Acrobat PubSec, showing run-time icon and tool tip overlays***


Inky Pinky

Digitally signed by Inky Pinky
DN: CN = Inky Pinky, C = US, O =
Pinky Pie Plates Pty
Location: San Jose, CA
Date: 2003.04.24 13:16:38 -07'00'

Author's Signature (Valid signature)
Icon and tool tip overlays

Standard AP dictionary and layers

A signature's appearance is contained in the N (Normal) attribute of the signature annotation's AP dictionary. The R (Rollover) and D (Down) attributes are not used. The appearance is contained in a number of XObjects that are assembled to create layers. These XObjects are pulled together with the Do page marking operator as described in "Assembling XObject layers" on page 8.

Signature appearances use two standard layers and can have a variable number of optional user-definable layers. Each layer is represented by its own XObject.

Note: Prior to Acrobat 6.0, signature appearances were manipulated at run-time in order to display the validity of the signature. The validity was shown as a graphic icon and with an additional, optional text message. The manipulated portions of the signature appearance were contained in layers n1, n3 and n4. Beginning with version 6, Acrobat does not maintain support for signature appearances that can be manipulated, though legacy signatures with these appearances may continue to display correctly. Use of layers n1, n3, and n4 is not recommended.

The standard XObject layers are as follows:

n0 — Background layer, as preset when creating the signature field, for example by using the Acrobat form creation tool. In the example appearances in "Example signatures" on page 7, this layer is blank.

n2 — Signature appearance layer, containing information about the signature. In the example Unverified signature, this is the layer that contains the line art for the handwritten signature and the text giving the name, date, reason and location of the signature. The content of this layer can be dynamically created when the signature is created, but thereafter remains static. Specifically, it remains static when the validity state is changed.

The layers are painted in sequence, beginning with n0.

Handlers can define their own custom layers. Custom layers must not use the names n0 through n4.

Assembling XObject layers

This section describes how an appearance is assembled with the top-level and second-level XObjects and the standard layers. Care should be taken to follow the recommendations for setting the matrix values to indicate unity transforms.

Top-level XObject

The AP dictionary's N attribute references a top-level XObject. This top-level XObject is necessary to properly resize signature appearances when these appearances are referred to by more then one signature field. For an example, see object 17 in the graphic "Signature appearance objects in a PDF file" on page 11.

The top-level XObject performs a `Do` on the second-level XObject as follows:

```
q 1 0 0 1 0 0 cm /FRM Do Q
```

The in-line cm matrix may change if the signature is resized. The Matrix is unity (x,y translation of zero, scale factor = 1), and the BBox is the current bounding box of the annotation.

The location of a signature within a document can have a bearing on its legal meaning. For this reason, signature fields never refer to more than one annotation. If more than one location is associated with a signature, the meaning may become ambiguous. Regardless, all signatures should include the top-level XObject.

Second-level XObject

The standard layer n0 appears by name in the Resource dictionary of the second-level XObject's dictionary. The Matrix of this XObject is unity and the BBox is that of the original annotation.

The XObject stream appears as follows:

```
q 1 0 0 1 0 0 cm /n0 Do Q
```

Layer n0

This layer renders the background and border of the annotation. The attributes that specify the background and border exist in the signature field dictionary. These attributes are not set if the signature is created on-the-fly by the plug-in. However, if Acrobat was used to create a blank signature field, these values are set.

The Acrobat forms HFT calls can help in the process of obtaining these values. For example, to extract border and background information, refer to `AFPDWidgetGetAreaColors` and `AFPDWidgetGetBorder`. Other Acrobat forms HFT calls create the n0 XObject. These API calls are documented in the *Acrobat and PDF Library API Reference*.

Note: For information on the DigSigHFT calls that can be used to create appearances, see "DigSigHFT appearance APIs" on page 11.

If you are using a blank n0 layer, use this call to obtain the blank object:

```
DigSigGetStdXObj ( cosDoc, DSBlankXObj ) ;
```

Layer n2

The handler can create any appearance for this layer. The layer is static. It is not changed when the validity state of the signature is changed.

All appearance handlers that render text honor the font type and color defaults that were set for the signature annotation. These defaults can be obtained by calling `AFPDFieldGetDefaultTextAppearance` as defined in the *Acrobat and PDF Library API Reference*

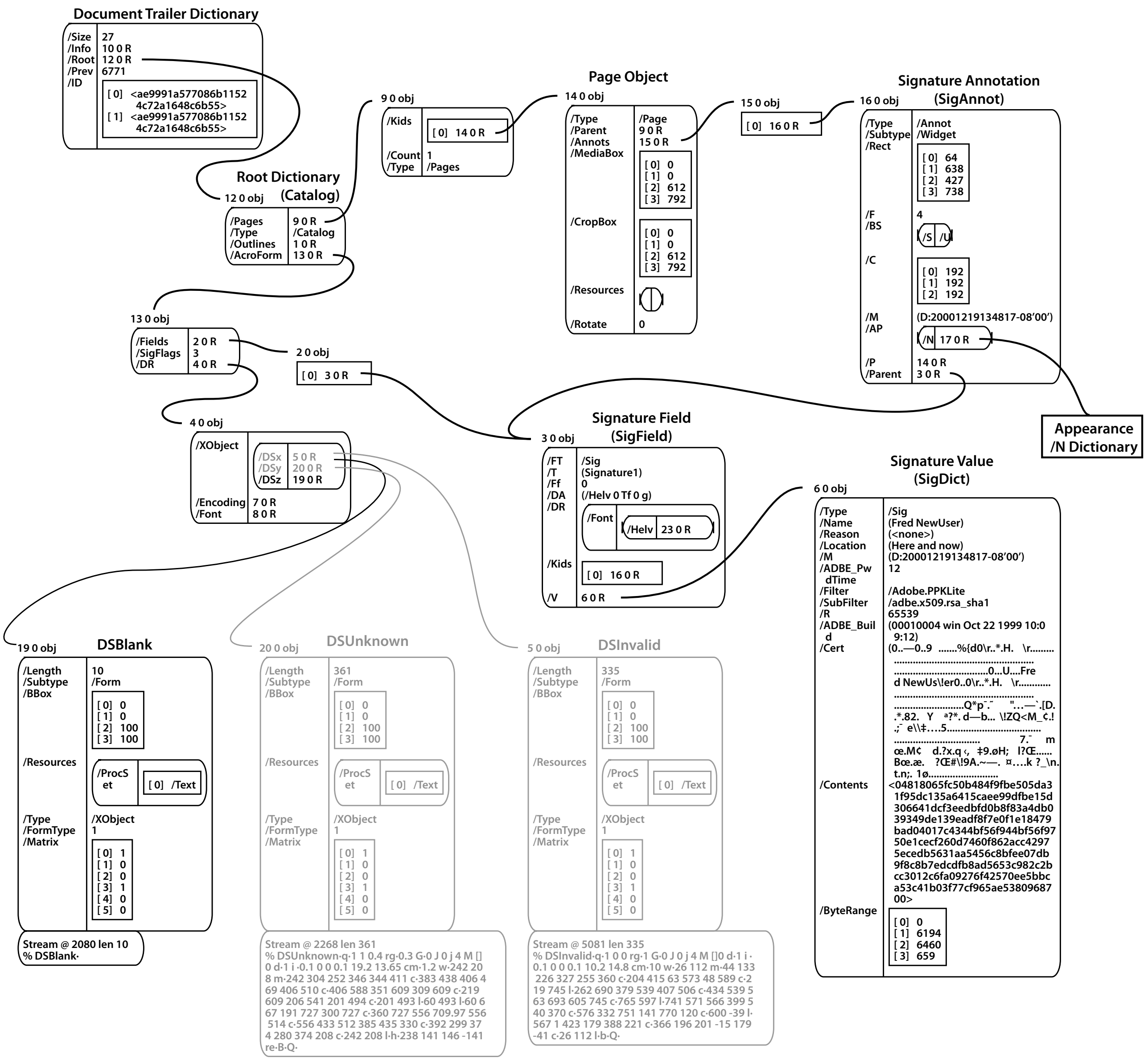
Signature annotation with no appearance

PDF signatures can be visible or invisible. In both situations there is an associated annotation. For invisible signatures, the annotation has a rectangle array with values [0 0 0 0]. This annotation is usually attached to the page that is being viewed when the signature is created. Despite not having an appearance, the annotation AP and N dictionaries may be present in some versions of Acrobat. If present, N references the `DSBlankXObj` (blank) XObject.

Signature objects

The following graphic shows the relationship of the signature objects in an example PDF file.

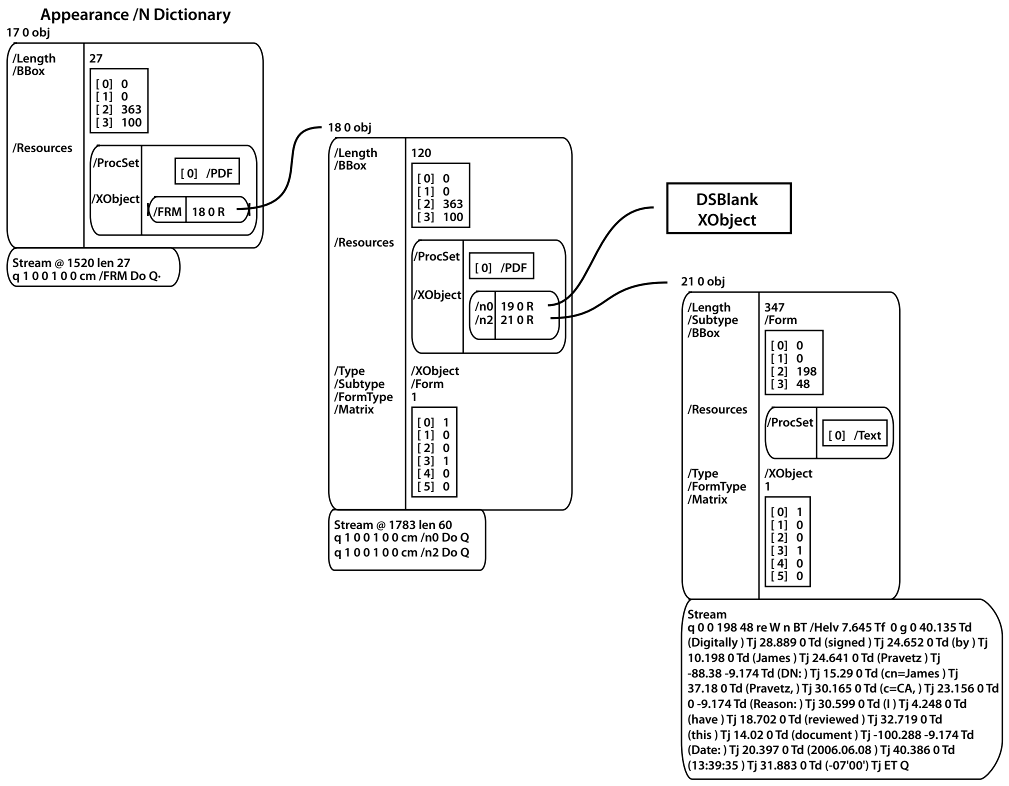
Signature objects in a PDF file


Document Trailer Dictionary
/Size 27
/Info 10 0 R
/Root 12 0 R
/Prev 6771
/ID
[0] <ae9991a577086b1152 4c72a1648c6b55>
[1] <ae9991a577086b1152 4c72a1648c6b55>
Root Dictionary (Catalog)
12 0 obj
/Pages 9 0 R
/Type /Catalog
/Outlines 1 0 R
/AcroForm 13 0 R
9 0 obj
/Kids [0] 14 0 R
/Count 1
/Type /Pages
Page Object
14 0 obj
/Type /Page
/Parent 9 0 R
/Annots 15 0 R
/MediaBox [0] 0 [1] 0 [2] 612 [3] 792
/CropBox [0] 0 [1] 0 [2] 612 [3] 792
/Resources
/Rotate 0
15 0 obj
[0] 16 0 R
Signature Annotation (SigAnnot)
16 0 obj
/Type /Annot
/Subtype /Widget
/Rect [0] 64 [1] 638 [2] 427 [3] 738
/F 4
/BS /S /U
/C [0] 192 [1] 192 [2] 192
/M (D:20001219134817-08'00')
/AP /N 17 0 R
/P 14 0 R
/Parent 3 0 R
Appearance /N Dictionary
13 0 obj
/Fields 2 0 R
/SigFlags 3
/DR 4 0 R
2 0 obj
[0] 3 0 R
4 0 obj
/XObject /DSx 5 0 R /DSy 20 0 R /DSz 19 0 R
/Encoding 7 0 R
/Font 8 0 R
Signature Field (SigField)
3 0 obj
/FT /Sig
/T (Signature1)
/Ff 0
/DA (/Helv 0 Tf 0 g)
/DR /Font /Helv 23 0 R
/Kids [0] 16 0 R
/V 6 0 R
Signature Value (SigDict)
6 0 obj
/Type /Sig
/Name (Fred NewUser)
/Reason (<none>)
/Location (Here and now)
/M (D:20001219134817-08'00')
/ADBE_PwdTime 12
/Filter /Adobe.PPKLite
/SubFilter /adbe.x509.rsa_sha1
/R 65539
/ADBE_Build (00010004 win Oct 22 1999 10:09:12)
/Cert (0..—0..9 ....%{d0\r..*.H. \r......... ....0...U....Fred NewUs\!er0..0\r..*.H. \r............ ....Q*p¯.˜ "…—`.[D. .*.82. Y ª?*. d—b... \!ZQ<M_¢.! .;¯ e\\‡….5.................................. .... 7.¯ m œ.M¢ d.?x.q ‹, ‡9.øH; l?Œ...... Bœ.æ. ?Œ#\!9A.~—. ¤….k ?_\n. t.n;. 1ø..........................
/Contents <04818065fc50b484f9fbe505da31f95dc135a6415caee99dfbe15d306641dcf3eedbfd0b8f83a4db039349de139eadf8f7e0f1e18479bad04017c4344bf56f944bf56f9750e1cecf260d7460f862acc42975ecedb5631aa5456c8bfee07db9f8c8b7edcdfb8ad5653c982c2bcc3012c6fa09276f42570ee5bbca53c41b03f77cf965ae5380968700>
/ByteRange [0] 0 [1] 6194 [2] 6460 [3] 659
19 0 obj DSBlank
/Length 10
/Subtype /Form
/BBox [0] 0 [1] 0 [2] 100 [3] 100
/Resources /ProcSet [0] /Text
/Type /XObject
/FormType 1
/Matrix [0] 1 [1] 0 [2] 0 [3] 1 [4] 0 [5] 0
Stream @ 2080 len 10
% DSBlank·
20 0 obj DSUnknown
/Length 361
/Subtype /Form
/BBox [0] 0 [1] 0 [2] 100 [3] 100
/Resources /ProcSet [0] /Text
/Type /XObject
/FormType 1
/Matrix [0] 1 [1] 0 [2] 0 [3] 1 [4] 0 [5] 0
Stream @ 2268 len 361
% DSUnknown·q·1 1 0.4 rg·0.3 G·0 J 0 j 4 M [] 0 d·1 i ·0.1 0 0 0.1 19.2 13.65 cm·1.2 w·242 208 m·242 304 252 346 344 411 c·383 438 406 469 406 510 c·406 588 351 609 309 609 c·219 609 206 541 201 494 c·201 493 l·60 493 l·60 667 191 727 300 727 c·360 727 556 709.97 556 514 c·556 433 512 385 435 330 c·392 299 374 280 374 208 c·242 208 l·h·238 141 146 -141 re·B·Q·
5 0 obj DSInvalid
/Length 335
/Subtype /Form
/BBox [0] 0 [1] 0 [2] 100 [3] 100
/Resources /ProcSet [0] /Text
/Type /XObject
/FormType 1
/Matrix [0] 1 [1] 0 [2] 0 [3] 1 [4] 0 [5] 0
Stream @ 5081 len 335
% DSInvalid·q·1 0 0 rg·1 G·0 J 0 j 4 M []0 d·1 i ·0.1 0 0 0.1 10.2 14.8 cm·10 w·26 112 m·44 133 226 327 255 360 c·204 415 63 573 48 589 c·219 745 l·262 690 379 539 407 506 c·434 539 563 693 605 745 c·765 597 l·741 571 566 399 540 370 c·576 332 751 141 770 120 c·600 -39 l·567 1 423 179 388 221 c·366 196 201 -15 179 -41 c·26 112 l·b·Q·

The XObjects DSUnknown and DSInvalid are shared XObjects that are used for run-time manipulation of signature appearances. These XObjects are used only for signature appearances with layers n1 and n3. Documents that do not contain these layers do not require these two layers to be present. DSBlank is used by layer n0. Acrobat 6.0 and later does not create signatures with layers n1 and n3.

The objects that contain the appearance for this example PDF file are shown in the following graphic.

Signature appearance objects in a PDF file



```
q 0 0 198 48 re W n BT /Helv 7.645 Tf  0 g 0 40.135 Td
(Digitally ) Tj 28.889 0 Td (signed ) Tj 24.652 0 Td (by ) Tj
10.198 0 Td (James ) Tj 24.641 0 Td (Pravetz ) Tj
-88.38 -9.174 Td (DN: ) Tj 15.29 0 Td (cn=James ) Tj
37.18 0 Td (Pravetz, ) Tj 30.165 0 Td (c=CA, ) Tj 23.156 0 Td
0 -9.174 Td (Reason: ) Tj 30.599 0 Td (I ) Tj 4.248 0 Td
(have ) Tj 18.702 0 Td (reviewed ) Tj 32.719 0 Td
(this ) Tj 14.02 0 Td (document ) Tj -100.288 -9.174 Td
(Date: ) Tj 20.397 0 Td (2006.06.08 ) Tj 40.386 0 Td
(13:39:35 ) Tj 31.883 0 Td (-07'00') Tj ET Q
```

DigSigHFT appearance APIs

Several API calls in the DigSigHFT make the generation of signature appearances easier for third-party developers. These procedures also make it easier to create signatures that conform to the layered XObject approach. The procedures are declared in DigSigHFT.h and documented in the *Acrobat and PDF Library API Reference*. The calls are:

`DigSigCreateStdXObj` — Creates a standard XObject from one of the values in the enum `DSXObjType`. This call allows a handler to get the predefined XObject for either a valid signature (a question mark that covers a pen) or a double valid signature (a check mark).

`DigSigAPCreateLayeredStream` — Creates the N dictionary and lower level XObjects given the XObjects for the various layers. The layers have defaults, so you do not need to define them.

`DigSigAPXObjectFromXObjList` — Creates a single XObject from multiple components. Use this routine to combine graphics and text components to make layer n2. Controls are given for placement of the graphics and text within the annotation.

`DigSigAPXObjectFromLogo` — Creates an XObject from a string that contains the marking operators needed to draw an object. Adobe’s signature handlers use this routine to create the Acrobat logo that is used as a signature watermark. As an example, the string that is used to create the logo is created by concatenating the strings shown in the following example.

`DigSigAPCreateCompositeTextXObj` — Combines various text elements into a single text XObject. This call calls `DigSigAPXObjectFromXObjList` to do the actual stitching of text elements.

Example 5 ***Example string to use with DigSigAPXObjectFromLogo***

```
// Precise bbox of Acrobat logo stream stmLogoLogo
const ASFixedRect CSSEngine::stmLogoBBox =
{ 0,0,0x00630000,0x00630000 }; // 0,0,99,99
// Stream data for Acrobat logo
const char* const CSSEngine::stmLogoData[] = {
"% Acrobat Logo",
"q",
"0 J 0 j 4 M []0 d",
"1 0 0 1 0 -1 cm",
"1 0.85 0.85 rg",/* faded red */
"0 i",
"96.1 24.7 m",
"96.4 24.7 l",
"96.8 24.7 97 24.6 97 24.3 c",
.
.
.
.
.
"51.4 45.2 60.3 39.4 65.1 36.6 c",
"53.5 34.6 40.1 31.1 28.1 25.9 c",
"h f",
"Q",
NULL
};
```

Appearance management with PubSec

The authoring of public key security handlers uses common signature appearance code in the PubSec software module. PubSec is a layer of code between the DigSigHFT and the PubSecHFT. Plug-in authors can allow PubSec to take care of all signature appearance generation and maintenance, but can continue to create their own signature appearances.

For more information on the PubSec API, see the *Acrobat and PDF Library API Reference*.

Appearance file

PubSec maintains a user-configurable database of signature appearances that can be used when signing a document. Users configure signature appearances using the Preferences settings in Acrobat. The PubSec signing dialog boxes access this same database, allowing users to configure and choose the signature appearance to be used. The database is a PDF file called appearances.acrodata that, for Windows, is stored in the Application Data section of a user's file system. An example path is:

C:\Documents and Settings\user\Application Data\Adobe\Acrobat\x.0\Security\appearances.acrodata

The PubSec signature appearance database is accessible via the `DSAPFile` interface of PubSecHFT. The `DSAPFile` interface exposes enough calls to allow maintenance of the database. Maintenance includes enumeration, addition, modification, and deletion calls.

`DSAPFileAcquire` — Acquires the `DSAPFile` object for PubSec's signature appearance database file. If the file has not already been acquired, it will be opened. PubSec acquires the database file whenever it is needed by PubSec, so handlers do not need to acquire this file unless it needs to access the file for other reasons.

`DSAPFileRelease` — Releases the file, with the potential to close and save the file.

`DSAPFileSave` – Saves the file if it is dirty, leaving the file open.

`DSAPFileGetCount` — Returns the number of configured appearance entries in the appearance database file.

`DSAPFileCanDeleteNthEntry` — Returns true if this entry in the `DSAPFile` can be edited (that is, is not read-only). Index values that are less then zero correspond to the default appearance, which is called the StandardText appearance in the user interface.

`DSAPFileGetNewNthName` — Gets a copy of the name of the Nth appearance object in the file. Use this when building a list of appearances for a user to choose from or edit.

`DSAPFileRemoveNthEntry` — Deletes the Nth appearance entry from the file.

`DSAPFileEditNthEntry` — Displays a user interface that allows the user to edit the Nth entry of the appearance file. Passing in indexes larger than the number of entries in the appearance file will create a new entry in addition to displaying the aforementioned user interface.

`DSAPFileCopyNthEntry` — Creates a copy of the Nth entry in the appearance file and appends the copy to the end of the list of signature appearances in the file. When copying default appearance entries, the resulting copy will not be considered a default appearance entry.

Annotation appearance

Signatures that are created by PubSec use the layered XObject approach.

Example 6 ***Signature appearance***

```
q 1 0 0 1 0 0 cm /n0 Do Q
q 1 0 0 1 0 0 cm /n2 Do Q
```

PubSec correctly displays legacy signatures that use the n1, n3, and n4 layers, but support of these layers in future versions of Acrobat may be discontinued at any time. At run time, PubSec changes these layers to show the validity state. PubSec looks for the presence of these layers to determine if run-time appearance modification is required. In some situations, PubSec will specifically warn against the presence of signatures with these layers because, for conformance with various signed document standards, signature appearances should be static.

Appearance watermarks

PubSec allows customization of the watermark that is shown as part of a signature appearance. Customization can be done by a PubSec handler or by a system administrator using the SignatureLogo.pdf file, which is described below.

The watermark is incorporated as part of the n2 layer of the signature appearance. In the example Signature appearance with a watermark created using Acrobat PubSec, layer n2 is composed of the watermark, which is the logo from the *US Department of the Interior*, the text "Digitally signed by ...", and the signer’s name, Inky Pinky.

To customize the watermark, a system administrator can place a file called SignatureLogo.pdf in the Security folder. If this file exists, the first page of this file is extracted and used for the watermark. If this file does not exist, PubSec handlers can provide the Acrobat logo as a watermark using the PubSec API `PSGetLogo` procedure call. If the `PSGetLogo` procedure is `NULL`, no logo watermark is added to the signature appearance.

The Security folder is the same folder where the appearances.acrodata file is stored. In Windows, an example path to the appearance file is:

C:\Documents and Settings\user\Application Data\Adobe\Acrobat\x.0\Security\appearances.acrodata

Creating a watermark

You can create a watermark from a vector or bitmap image file.

➤ **To create a watermark from a vector-based drawing:**

1. Open a file containing the design (for example, a company logo) using Adobe Illustrator®.
2. Isolate the design and copy it to the clipboard.
3. Create a new document and paste the design into this new document.
4. Select all. Note that if your logo has objects that are drawn over top of other objects, the underlying objects may become visible through the higher layers. In this situation you may have to group the overlapping objects before proceeding to the next step.
5. Click **Window** > **Transparency** and drag the transparency slider to approximately 20%. Note that you want the appearance to be sufficiently subdued, so a value of less than 20% may be necessary for some designs.
6. Click **Object** > **Flatten Transparency**, set the Raster/Vector balance to 100%, and click OK.
7. Save the file as a PDF document.
8. Open the PDF file in Acrobat Professional.
9. Click **Document** > **Crop Pages**. Check the remove white margins option and click OK.
10. Save the file in an appropriate location.
11. Test the file (see “Testing or correcting a watermark file” on page 15).

➤ **To create a watermark from a bitmap image, such as a JPEG file:**

1. Drag the bitmap image file to Acrobat to create a new PDF file.
2. Save the PDF file.
3. Prepare the file using the commenting stamp tool (see “Testing or correcting a watermark file” on page 15).

Testing or correcting a watermark file

After you have created a watermark, you can test it or correct it if it causes problems.

➤ **To test a watermark PDF file:**

1. Copy or save your watermark PDF file as SignatureLogo.pdf in the Security folder.
2. In Acrobat, click **Edit > Preferences > Security**.
3. Click **New** to create a new signature appearance. Your watermark should appear in the Preview box. Set the desired options, click OK, and click OK again.
4. Try signing a document to verify that the watermark is working. If you receive an error when testing a document (or if the watermark fails to appear in the Preview box), see To Correct the Structure of a Watermark File.

Not all PDF files can be used as signature watermarks. If the structure of the first page is not correct, you can correct it with the Stamp tool in Acrobat.

➤ **To correct the structure of a watermark file:**

1. Note the contents of the Stamps folder to ensure you do not overwrite any existing stamps. This folder is at the same level as the Security folder, for example:

   C:\Documents and Settings\user\Application Data\Adobe\Acrobat\x.0\Stamps
2. In Acrobat, click **Tools** > **Commenting** > **Stamps** > **Create Custom Stamp**. Select your watermark file and import it as a new stamp in a new category.
3. Copy the new file that has been added to the Stamps folder to the Security folder and rename it as SignatureLogo.pdf.
4. Test the file (see the preceding procedure).

Adobe does not provide support for logo watermark preparation or use. If you have trouble with any aspects of this procedure, consult a graphics professional.