Digital Signatures Workflow Guide

A guide for workflow owners

For the Acrobat family of products

Acrobat® Family of Products
Modification date: 9/28/12

Digital Signatures Workflow Guide for the Adobe® Acrobat Family of Products.

Contents

1 Getting Started

This guide contains information about the Acrobat family of products' digital signature feature. Unlike the 9.x guide, it does not contain information about document security methods (password, certificate, and ALCRMS security) or end user details.

Readers should also note that some legacy and esoteric information has been migrated to the *Addendum Digital Signatures Guide*. These changes are designed to reduce the scope and number of pages in this document, thereby improving it's usability. The information removed from this guide continues to reside in the Content Security Library and includes the following:

- *Addendum to the Digital Signatures Guide*: Contains legacy information, technical details, and end user instructions not necessarily required for configuring the product.
- *Document Security Guide*: Contains information about password security, certificate security and Adobe LiveCycle Rights Management Server security.

Audience

While this document does contain some end user "how to use the feature" content, it primarily provides details not found in end user help or in the SDK. The primary focus here is to help administrators and other business users set up and maintain secure PDF workflows. Potential audiences might include:

- Administrators who configure and deploy applications across their enterprise.
- Developers who need registry level detail to augment SDK information about creating custom plug-ins and handlers that use Acrobat's security features.
- Other business users that need advanced knowledge of signature workflows.

1.1 What's changed with 10.1

- Acrobat/Reader 9.4.5 and Acrobat/Reader 10.1 will accept Identrust certificates with a SHA256 hash. Prior to these releases, certificates with OIDs under the Identrust arc enforced the use of SHA1.
- The Signature Properties dialog provides a clearer presentation of signing time and validation times:
  - The Date fields on Date/Time and Summary tabs have been renamed to Signing Time. The field values always contains both the date and the signing time from the signer's computer.
  - The Date/Time and Summary tabs contain more validation time details. In addition to the signing time from the signer's computer, the validation time also appears in some workflows. For example, when a timestamp is present (including post-signing timestamps), the Date/Time tab displays timestamp details such as if it is embedded, the timestamp authority name, and other information. Some of this additional information also appear in the Certificate Viewer.
- Prior to 10.1, OCSP responses without nextUpdate were never embedded in a signature. For 10.1 and later, OCSP responses are always embedded irrespective of the presence of nextUpdate; however, whether they are used for signature validation depends on certain conditions:
  - Validation time is greater than thisUpdate minus the value of maxClockSkew (the default is 5 minutes). This test is always performed.

- When nextUpdate is present and the validation time is less than the nextUpdate time plus the value of maxClockSkew.
- When nextUpdate is not present and the validation time is less than the thisUpdate time or the producedAt time (whichever is greater) plus the value of maxClockSkew.

If you need a relaxed security environment (for example, when the responder is caching OCSP responses), bIgnoreNextUpdate can be set to 1 to ignore the last test. In this case, embedded responses without nextUpdate are always used for signature validation provided that they pass first test.

Note: Note: This behavior is designed to support Acrobat's long term validation feature and allows validating a signature with embedded responses that were valid at signing time.

- Portfolio Signature Status indicators: 10.1 implements a portfolio status indicator called a Signature Badge. A signature badge button appears in a signed PDF portfolio window's taskbar except when the portfolio's cover sheet is visible. Like the signature in the Signature panel, a status icon shows the signature status.

Note: Although the portfolio user interface can give the impression that the cover sheet is some kind of special attachment to a portfolio document, the cover sheet is the same as the top-level portfolio document. The signature status of the portfolio's cover sheet is the same as signature status of the entire portfolio.

The badge has the following behaviors:

- The portfolio document's HideToolbar viewer preference is always honored even if it means that the badge will not be initially visible when the document is opened. The user can hide or show the taskbar using **View > Show/Hide > Toolbar Items > Show/Hide Toolbars** menu.
- Clicking the badge button displays the cover sheet as well as the Signatures panel.
- The badge displays a status icon which is similar to other status icons. For those states that display the common name, it is gotten from the "cn" key in the Subject section of the signing certificate.

1.2 What's changed with 10.0

The following enhancements have been introduced with 10.0:

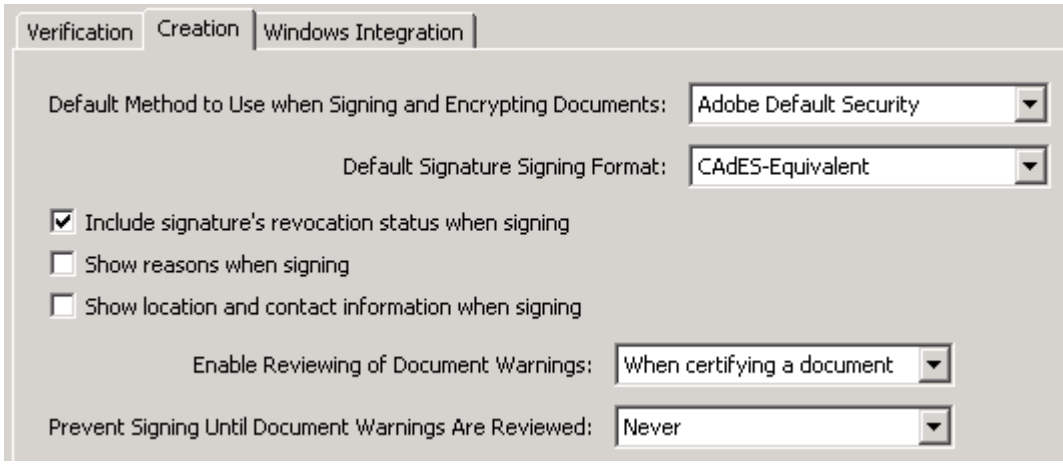
- **Content security menus**

  Many of the menu items have been migrated to a new Tools menu bar which can be hidden or displayed as desired. Top level menu items have changed as follows:

  - **Acrobat menu changes**: To get the content security menu items, choose **Tools**.
    - The Security Settings Console, Import/Export Security Settings, and document security settings are under **Tools > Protection > More Protection**.
    - Signature related settings (expect for Digital ID settings which are in the Security Settings Console) are under **Sign and Certify**.
    - The Trusted Identity Manager is under **Tools > Sign and Certify > More Sign and Certify**.
    - The default signature format can be selected from the **Preferences > Security > Advanced Preferences > Creation tab**.

    - The **Show timestamp warnings in Document Message Bar** has been removed from the **Preferences > Security > Advanced Preferences > Verification tab**.
  - **Reader menu changes**: Content security menu items now reside under **Edit > Protection**.
- **More nonce options**: `bSendNonce` is deprecated and replaced by `iSendNonce`. The additional configuration option has been added which allows nonces but still permits valid signatures if they do not exist.
- **Long term validation and timestamping**:
  - **Timestamps**: The default size set aside for timestamps increased from 4096 to 6144. This value is controlled by the `iSize` key in the registry.
  - **Invisible timestamping**: Previous versions of Acrobat only allowed timestamps as part of a signature workflows. It is now possible to timestamp a document without going through the standard signing workflow (The document is still signed). There is a separate menu item called **Time Stamp Document**.
  - **New default signing format choice**: ETSI.CAdES.detached is now a supported value for `aSignFormat`, and it can be added via enterprise preference configuration (e.g. registry) or via the signature creation preferences in the user interface.
  - **Support for the PAdES and CAdES standards**: Such support is part of PDF's enhanced long term validation and timestamping capabilities (see above). For details about the standards, see PDF Advanced Electronic Signatures (PAdES), Parts 1,2,3 and 4 (http://pda.etsi.org/pda/queryform.asp) and ETSI/ESI Technical Standard (TS) 102 778 (http://pda.etsi.org/pda/queryform.asp).

Figure 1 CAdES signing format in UI



- **Non-explicit OID support**: Acrobat products now support non-explicit OIDs with 10.0. Support for the non-explicit model allows Acrobat products to conform to the processing model used by many European Qualified CAs, SAFE, and others in which only the signing certificate (end entity) is checked against the initial-policy-set. The user interface
- **PKCS#11 tokens and smartcards**: Acrobat continues to provide robust device support. However, this feature interacts with the new Protected Mode in Adobe Reader in a way which may require a couple of extra installation steps. If you encounter a problem installing a PKCS#11 driver, disable Protected Mode, install the driver, and then re-enable Protected Mode.

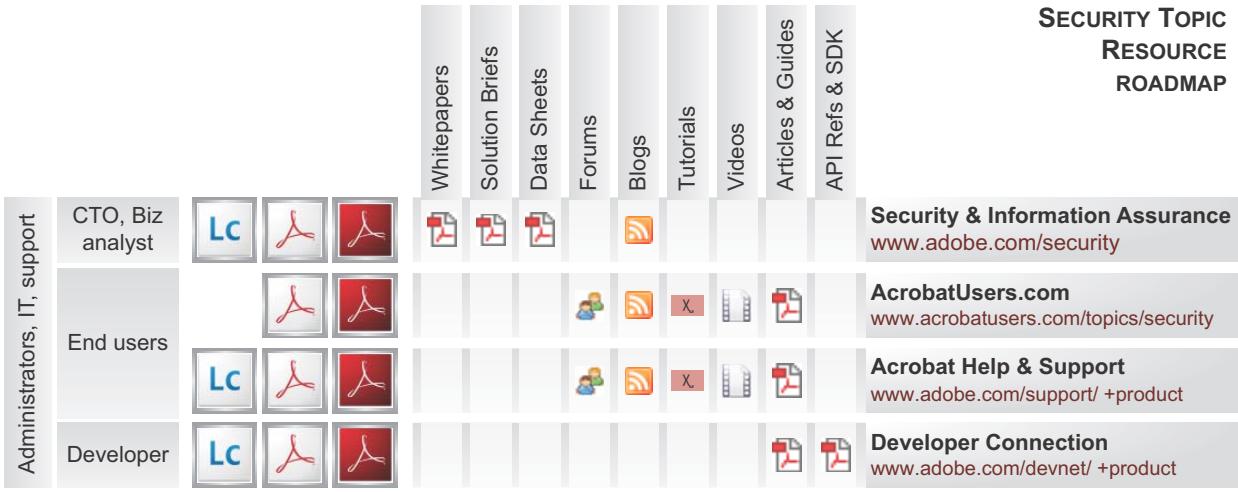
- **Adobe Database Connectivity (ADBC) support is removed**: The ability to connect to Windows ODBC (Open Database Connectivity) data sources is deprecated for security reasons.
- **SDK and API extensions**: For information about new APIs and resources, download the SDK and its documentation.

1.3 Other resources

This guide provides details that are probably not of interest to the casual user. It is also not a developer document, and developers should refer to the SDK and its associated references and APIs. As you peruse this document, keep in mind that there are numerous resources out there, including forums and even video tutorials. Adobe is aggressively revamping many of its learning resources as Web 2.0 matures, and it may be that one of these sites would prove equally, if not more, useful:

- Enterprise Documentation: Content Security Library
- Developer documentation: http://www.adobe.com/go/acrobat_security.
- End user documents:
  - http://www.adobe.com/support/acrobat: An evolving resource containing tutorials, guides, videos, blogs, and other help.
  - http://www.adobe.com/support/livecycle
  - http://www.adobe.com/support/reader
- White papers/data sheets: http://www.adobe.com/security
- www.acrobatusers.com: Provides moderated forums, tutorials, and videos.

Figure 2 Resource roadmap



Note: Table 1 shows a partial list of the documentation residing at the above locations.

Table 1 Documentation related to security

Document	Audience	For information about
Acrobat SDK Documentation Roadmap	Developers	A guide to the documentation in the Adobe Acrobat SDK.
Acrobat and PDF Library API Reference	Developers	A description of the APIs for Acrobat and Adobe Reader® plug-ins, as well as for PDF Library applications.
JavaScript for Acrobat API Reference	Developers	A listing of the Acrobat JavaScript APIs.
Developing Acrobat Applications with JavaScript	Developers	Additional detail about the Acrobat JavaScript APIs.
PDF Reference 1.x	Developers	A detailed description of the PDF language.
FDF Data Exchange Specification	Developers	A object-level FDF file description. The files can be generated programmatically and used to share security-related data.
PDF Signature Build Dictionary Specification	Developers	Build properties for the PDF Reference's signature dictionary which provides interoperability details for 3rd party handlers.
Digital Signature Appearances	Developers & administrators	Guidelines for creating signatures programmatically.
Guidelines for Developing CSPs for Acrobat on Windows	Developers & administrators	Guidelines for developing a Cryptographic Service Provider for use with Acrobat® on the Windows® platform.
Enhanced Security in Adobe Acrobat and Adobe Reader	Administrators & end users	Details about protecting signing environments by controlling document trust at the file, folder, and host levels.
Digital Signatures in the PDF Language	Developers	A generic description of how signature work in PDF.
Digital Signatures in Acrobat	Anyone needing an overview	A description of how signatures are implemented in Acrobat.

2 Authoring Signable Documents

Acrobat's digital signature capabilities allow authors to set up a secure signing environment and create simple documents and complex forms with one or more fields. Document authors can design documents with multiple signature fields each with unique behavioral characteristics and appearances. A signed field can lock other fields so that signed data can't be changed, and authors can force certain signature fields to be a required part of a workflow. Attention to signature field design and configuration can help you make the document "do the right thing" when someone receives it as well as control what that person can and cannot do with it.

For more information, refer to the following:

- "Best Practices for Signed Documents that will Change" on page 11
- "Setting up the Signing Environment" on page 11
- "Working with Signature Fields" on page 22
- "Authoring Signable Forms" on page 27

2.1 Best Practices for Signed Documents that will Change

Some workflows require that someone enter form data, provide annotations and comments, or sign a document. When the form author has authorized or the application allows such changes, the changes are not flagged as problematic or warnings. In general, the goal should be to design documents and workflows so that both the signature status and document status are always valid.

Best practices for signed documents that will change vary by role:

- Document author: Form fields can be ordered, named, and associated with behaviors that limit changes in signing workflows.
- **First signer**: Use a certification signature for the first signature in a document and set **Permitted Changes after Certifying** as needed. The specified actions should not result in a warning triangle to appearing on signatures.
- **Signature validators and subsequent signers**: View the signed version of the document and look at the signature's status icon in the Signature's pane as well as the document status icon in the message bar. If there are any issues or problems, read the text. You may also wish to view the document the PDF signature report, view modifications, and so on.

2.2 Setting up the Signing Environment

A number of preferences control how your application, the document, and the signature will behave in signing workflows. These preferences tell the application where to look for Windows certificates, control signature appearances, enable the use of preview mode, and so on.

Tip: Participants in signing workflows (both document authors and signers) should review their application settings and configure their environment. Some preferences control authoring, some only have to do with signing, and some impact both.

Table 2 Signing environment preferences

Importance	Description	Section
Recommended	A number of preferences affect the signing workflow and resulting signature. All users should review these settings.	Setting Signing Preferences
Optional	Create default user information ahead of time to save time later.	
Optional	Use the default signature appearance or create your own.	Signature Appearance Configuration
Optional	To sign with a timestamp, configure a timestamp server.	Timestamps for Signing

2.2.1 Setting Signing Preferences

Preferences options vary by platform and application as follows:

1. Choose one of the following:
   - Acrobat (Windows): **Edit > Preferences > Security**
   - Acrobat (Macintosh): **Acrobat > Preferences > Security**
   - Adobe Reader (Windows): **Edit > Preferences > Security**
   - Adobe Reader (Macintosh): **Adobe Reader > Preferences > Security**
2. Set your preferences as described in the following sections:
   - "Requiring Preview Mode" on page 12
   - "Changing the Default Signing Method" on page 13
   - "Embedding Signature Revocation Status" on page 14
   - "Allowing Signing Reason" on page 15
   - "Showing Location and Contact Details" on page 15
   - "Enabling Document Warning Review" on page 15
   - "Requiring Document Warning Review Prior to Signing" on page 16
   - "Enabling a Warnings Comment or Legal Attestation" on page 16

2.2.1.1 Requiring Preview Mode

In general, everyone within a signing workflow needs to know that what they are viewing is actually what is signed or being signed. Preview mode automatically checks document integrity and generates a report that itemizes any content that could potentially prevent the signer from knowing what they are signing.

Preview mode provides several benefits:

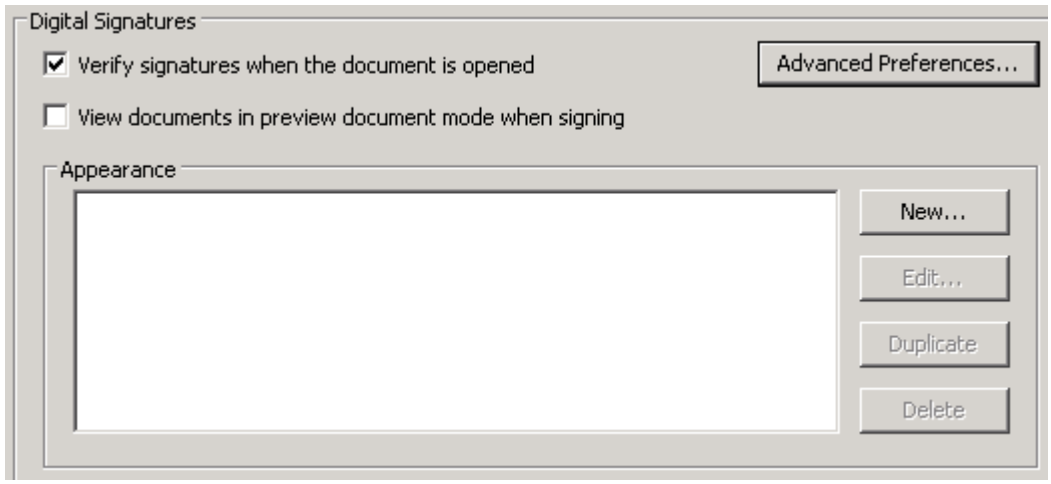
- It checks the document for elements that may prevent a signer from seeing what they are signing.

- It suppresses document elements that may prevent a signer from knowing what they are signing.
- It generates a report about those elements. For details, see Chapter 4, "Document Integrity and Preview Mode".

To use preview mode automatically:

1. Choose one of the following:
   - Acrobat (Windows): **Edit > Preferences > Security**
   - Acrobat (Macintosh): **Acrobat > Preferences > Security**
   - Adobe Reader (Windows): **Edit > Preferences > Security**
   - Adobe Reader (Macintosh): **Adobe Reader > Preferences > Security**
2. Set **View documents in preview document mode when signing**.

Figure 3 Preview document mode preference



2.2.1.2 Changing the Default Signing Method

In some enterprise situations administrators may require a method other than Adobe Default Security. For example, non-Adobe plugins may be used in business environments that require support of biometrics, signature escrow, alternative methods of private key access, and so on. In those cases, administrators may specify an alternate plugin or provide user training on how to choose the right one.

Third party plugins include:

- **Entrust® plug-in for Acrobat 4 and 6**: This plugin interfaces to the Entrust Entelligence desktop application and provides the same functionality that is provided by Adobe's plugin. Businesses that use Entrust for PKI deployment may require the Entrust plug-in.
- **SignCube® plug-in for Acrobat 7**: The SignCube plugin is used to create signatures recognized as valid under the German Digital Signature Law.
- **CIC**: The Communication Intelligence Corporation® Plugin (CIC) is used by some banks and insurance companies to provide an electronic version of handwritten signatures. This plugin limits users' ability to use encryption.

To change the default signing method:

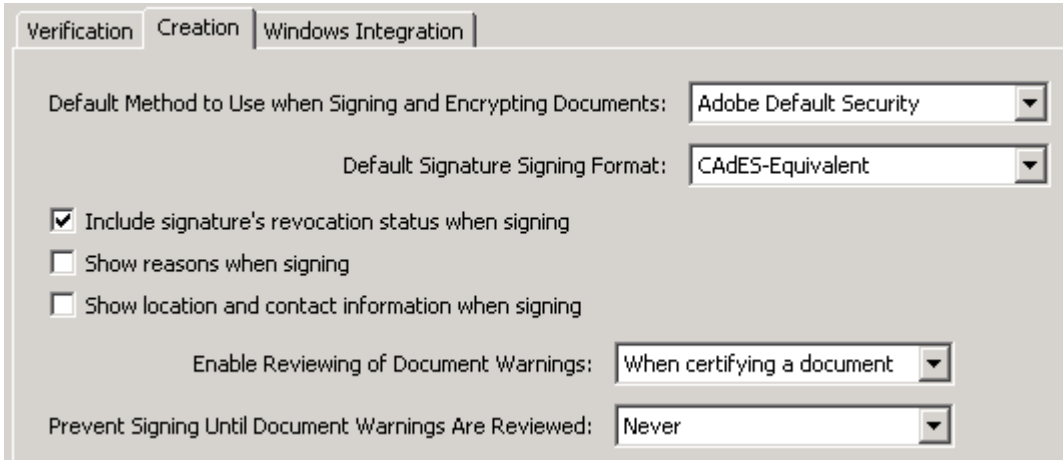
1. Choose one of the following:
   - Acrobat (Windows): **Edit > Preferences > Security**
   - Acrobat (Macintosh): **Acrobat > Preferences > Security**
2. Choose **Advanced Preferences**.
3. Choose the Creation tab (Figure 4).
4. Under **When Verifying**, check one of the last two radio buttons so that the **Default Method for Verifying Signatures** drop down list becomes active.
5. Select a signing method.

   Note: Do not change the signing method unless instructed to do so by your administrator.

2.2.1.3 Embedding Signature Revocation Status

1. Choose one of the following:
   - Acrobat/Reader (Windows): **Edit > Preferences > Security**
   - (Macintosh): **Acrobat (or Reader) > Preferences > Security**
2. Choose **Advanced Preferences**.
3. Choose the Creation tab (Figure 4).

Figure 4 Signature creation preferences



4. **(*Recommended*) Set Include signature's revocation status when signing**.

   Embedding the signing certificate's revocation status in a document allows recipients to validate certificates (signatures) while offline and speeds up the revocation checking process. Moreover, if a certificate is revoked or expired at some time after signing, embedded revocation information enables the application to determine if a certificate was valid at the time of signing so that the signature status will remain valid.

Note: Revocation checking occurs immediately after signing as well as during signature validation. If the revocation status is not embedded in a signature, the application looks in the locally stored certificate revocation list cache. If it is not there, the application goes online to complete the check.

2.2.1.4 Allowing Signing Reason

Turning this option on results in a **Reasons** field appearing in the signing dialog. The signer can then choose a default reason such as "I have reviewed this document" or create a new one.

1. Choose one of the following:
   - Acrobat (Windows): **Edit > Preferences > Security**
   - Acrobat (Macintosh): **Acrobat > Preferences > Security**
   - Adobe Reader (Windows): **Edit > Preferences > Security**
   - Adobe Reader (Macintosh): **Adobe Reader > Preferences > Security**
2. Choose **Advanced Preferences**.
3. Choose the Creation tab (Figure 4).
4. Set **Show reasons when signing**.

2.2.1.5 Showing Location and Contact Details

When this option is turned on, the **Location** and **Contact Info** fields appear in the signing dialog.

1. Choose one of the following:
   - Acrobat (Windows): **Edit > Preferences > Security**
   - Acrobat (Macintosh): **Acrobat > Preferences > Security**
   - Adobe Reader (Windows): **Edit > Preferences > Security**
   - Adobe Reader (Macintosh): **Adobe Reader > Preferences > Security**
2. Choose **Advanced Preferences**.
3. Choose the Creation tab (Figure 4).
4. Set **Show location and contact information when signing**.

2.2.1.6 Enabling Document Warning Review

Enabling document warning review allows signers to check document integrity prior to signing. The document can be analyzed to determine if it contains any content that could adversely impact the integrity of the signing process. For example, a document could contain JavaScript that could change a data field before or after a signature is applied. Setting an option here affects what appears in the **Prevent Signing Until Document Warnings Are Reviewed** drop-down list.

Note: **Enable Reviewing of Document Warnings** and **Prevent Signing Until Document Warnings Are Reviewed** settings function in tandem and should be set together. Setting both these options to *Always* results in the highest degree of assurance that the signing process is not adversely impacted by malicious content.

1. Choose one of the following:
   - Acrobat (Windows): **Edit > Preferences > Security**
   - Acrobat (Macintosh): **Acrobat > Preferences > Security**
   - Adobe Reader (Windows): **Edit > Preferences > Security**
   - Adobe Reader (Macintosh): **Adobe Reader > Preferences > Security**
2. Choose **Advanced Preferences**.
3. Choose the Creation tab (Figure 4).
4. Set **Enable Reviewing of Document Warnings**. Select from the following:
   - **Never**: Turns off document warning review. No **Review** button appears in the signing dialog.
   - **When certifying a document**: The **Review** button appears in the signing dialog only when a certification signature is being applied. This option allows the signer to add a legal attestation.
   - **Always**: The **Review** button always appears in the signing dialog. This option allows the signer to add a legal attestation for certification signatures.

2.2.1.7 Requiring Document Warning Review Prior to Signing

If document warning reviews are critical to your signing workflow, you can require them.

> **Note:** **Enable Reviewing of Document Warnings** and **Prevent Signing Until Document Warnings Are Reviewed** settings function in tandem and should be set together. Setting both these options to *Always* results in the highest degree of assurance that the signing process is not adversely impacted by malicious content.

1. Choose one of the following:
   - Acrobat (Windows): **Edit > Preferences > Security**
   - Acrobat (Macintosh): **Acrobat > Preferences > Security**
   - Adobe Reader (Windows): **Edit > Preferences > Security**
   - Adobe Reader (Macintosh): **Adobe Reader > Preferences > Security**
2. Set **Prevent Signing Until Document Warnings Are Reviewed**. Select from the following:
   - **Never**: Signing can continue without a document warning review.
   - **When certifying a document**: Signers must choose **Review** to apply a certification signature.
   - **Always**: Signers must always choose **Review** when signing.

2.2.1.8 Enabling a Warnings Comment or Legal Attestation

For certified documents that contain multimedia, comments, or other dynamic content, it often beneficial to add a warnings comment or legal attestation that states that the content is OK and permitted by the author. If document warnings are enabled, then the signer can review the warnings and either choose from Acrobat's default comment "I have included this content to make the document more dynamic," or create a custom comment.

To enable warnings comments on certified documents:

1. Set **Enable Reviewing of Document Warnings** to **When certifying a document** or **Always** as described in "Enabling Document Warning Review" on page 15.

2.2.2 Signature Appearance Configuration

The signing workflow allows you to select from a list of signature appearances. You can use the default appearance created from your name or create one or more custom appearances and store them for later use. An appearance consists of three main components, and each can be separately customized:

- **Signature**: The text or graphic that identifies the signer on the left hand side of the appearance. Photos and scanned signatures are common. When using a graphic, make the background transparent if a watermark should be visible in the underlying layer.
- **Watermark or background**: A background image or text that is automatically applied to each signature. By default, this is the Adobe PDF logo.
- **Signature details**: Signature data that the signer want to include in the appearance.

Figure 5 Custom signature appearance

2.2.2.1 Custom Signature Appearance

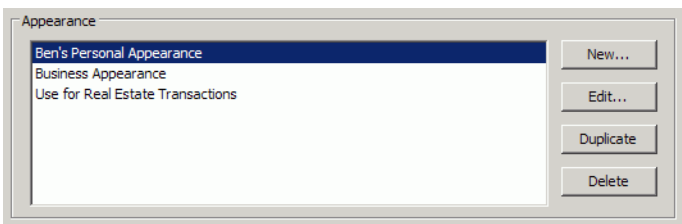
Users generally customize one or more signature appearances and store them for later use. Available signatures are listed in the appearance panel (Figure 6).

Note: If you have created a watermark file as described in "Custom Watermark and Backgrounds" on page 19, the watermark should automatically appear in all of your signature appearances.

To create a new signature appearance:

1. Choose **Edit > Preferences** (Windows) or **Acrobat > Preferences** (Macintosh).
2. Choose **Security** in the left-hand list.
3. In the Appearance panel, choose **New**.

Figure 6 Signature appearance: New button



4. Configure the signature appearance options:
    - **Title**: Any arbitrary title used to identify the appearance.
    - Set the graphic options in the Graphic panel
        - **No graphic**: No graphic is used.
        - **Imported graphic**: Choose **File > Browse**, select a file and choose **OK**.
        - **Name**: Your text name will appear instead of a graphic. The name is extracted from the signing certificate.

    Note: By default, the signature watermark is the Adobe PDF logo but it can be customized. To avoid obscuring a background, use line art with a transparent background.

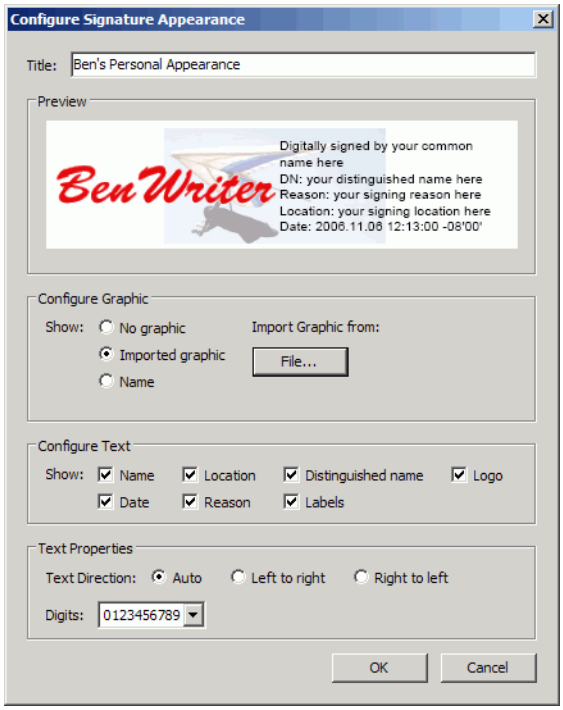
    - Set the text options in the Configure Text panel
        - **Name**: The name associated with the certificate.
        - **Date**: The date signed.

    Note: Signature appearances can only display local (computer) time, and it will likely differ from that in the Date/Time tab on the Signature Properties dialog when a timestamp server is used.

        - **Location**: The location associated with the identity configured in Acrobat.
        - **Reason**: The reason for signing.
        - **Distinguished name**: A name with details such as country, organization, organizational unit, and so on.
        - **Labels**: A label for each of the items above. For example, `Reason`.
        - **Logo**: The logo or graphic used as a background watermark.
    - Set the text options in the Text Properties panel
        - **Text Direction**: Choose a direction appropriate for the signer's language.
        - **Digits**: If languages are installed that use digits other than 1234567890, the drop-down list will be populated with alternate choices. Choose a digit set appropriate for the signer's language.

5. Choose **OK**.

Figure 7 Signature appearance: Configuration



2.2.2.2 Custom Watermark and Backgrounds

A watermark is a partially transparent graphic or logo that appears "behind" a signature. By default, the watermark is the Adobe PDF logo. Line (vector) art that is simple and unobtrusive often works best.

1. Import a logo or create a new one in a program such as Adobe Illustrator (used in the example below).
2. Set a low transparency level and flatten the transparency:
    1. Select all and group the objects if there is more than one.
    2. Choose **Window > Transparency** and slide the transparency slider to some low value such as 20%.
    3. Choose **Object > Flatten Transparency**. Leaving the Raster/Vector balance at 100%.
    4. Save the file to a PDF file.
3. Open the PDF file in Acrobat.
4. Crop the page and remove white space.

    Note: The method varies across product versions. For example, for 8.x , choose **Document > Crop Page** and check **Remove White Margins**.

5. Save the file as SignatureLogo.pdf in:

- **Windows**: C:\Documents and Settings\<user>\Application Data\Adobe\Acrobat\<version>\Security.
- **Macintosh**: \Users\<user name>\Library\Application Support\Adobe\Acrobat\<version>\Security.

2.2.2.3 Editing or Deleting a Signature Appearance

To edit a signature appearance:

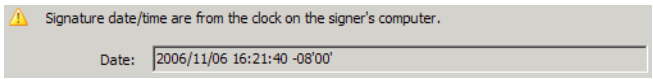
1. Choose **Edit > Preferences** (Windows) or **Acrobat > Preferences** (Macintosh).
2. Choose **Security** in the left-hand list.
3. Highlight an appearance in the Appearance panel.
4. Choose **Edit** or **Delete**.
5. Edit the appearance.
6. Choose **OK**.

2.2.3 Timestamps for Signing

Signature times tell you that a document and signature existed prior to the indicated time. All signatures are associated with the signer machine's local time, but they may also include a timestamp time provided by a timestamp server if one is configured. Because a user can set that time forward or back, a local time is less reliable than a timestamp time. Local times are labeled as such in the Date/Time and Summary tabs of the Signature Property dialog (Figure 8).

Note: Because signature appearances only display local time, the appearance time will be different from the timestamp time shown in the Date/Time tab of the Signature Properties dialog.

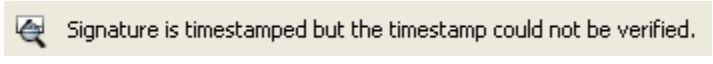
Figure 8 Timestamps: Local, machine time



Like signatures, timestamps are provided by someone (a timestamp authority) who uses certificates to confirm their identity. Before you can validate a timestamp, you must explicitly trust the timestamp authority's certificate. Timestamp certificate status appears in the Date/Time and Summary tabs of the Signature Property dialog:

- Untrusted timestamp certificates appear as follows:

Figure 9 Timestamps: Untrusted stamp



- Trusted timestamps that have been added to the Trusted Identities list and have been explicitly trusted for signing appear as follows:

Figure 10 Timestamps: Trusted stamp

Timestamps are usually provided by third-party timestamp authorities such as GeoTrust. Because timestamp authorities may charge for their services, Acrobat does not automatically set a default timestamp server if multiple servers are listed. Users must manually specify which timestamp server to use as the default.

Configuring Acrobat to use a Timestamp Server

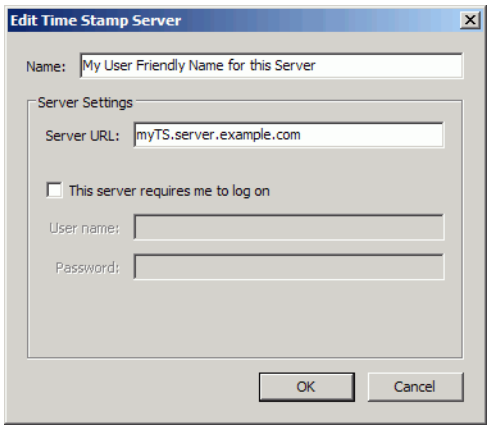
To use a timestamp when you sign, configure your application to use a timestamp server, set it as the default, and set trust the certificate of the timestamp authority. The timestamp server is always used if a default timestamp server is specified.

To manually set up a timestamp server:

1. Choose **Advanced** (Acrobat) or **Document** (Reader) **> Security Settings**.
2. Choose **Timestamp Servers**.
3. Choose **New**.

   In most cases, administrators preconfigure end user machines.
4. Enter the server settings:
   - **Name**: The server name.
   - **Server URL**: The server URL.
   - **Username**: The login username if required.
   - **Password**: The login password if required.
5. Choose **OK**.
6. Make this server or some other server the default by choosing **Set Default**. Timestamping cannot occur unless a default server selected.

Figure 11 Timestamps: Entering server details



2.3 Working with Signature Fields

Signature fields are a type of form field, and both Acrobat and Adobe Reader ignore whether they are authored with Forms Designer or Acrobat. Digital signatures behave uniformly irrespective of the authoring mechanism.

For details about customizing one or more fields, see the following:

- Specifying General Field Properties
- Customizing Field Appearances
- Changing the Default Field Appearance
- Cut, Copy, and Paste Signature Fields
- Arranging Signature Fields
- Creating Multiple Copies of a Signature Field
- Authoring a Document with Multiple Fields
- Locking Fields Automatically After Signing
- Unlocking a Field Locked by a Signature
- Making a Field a Required Part of a Workflow
- Specifying a Post-Signing Action

2.3.1 Creating a Blank Signature Field

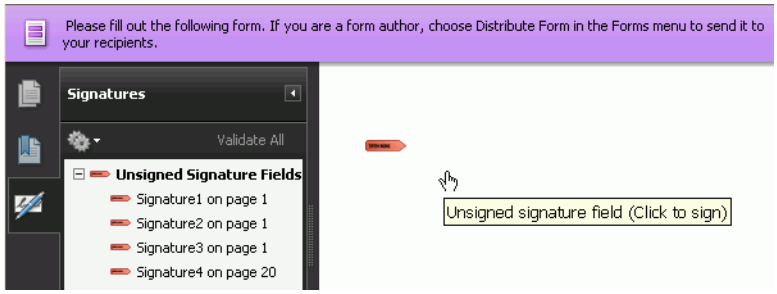
Signatures and related information are stored in a signature field embedded on the page. A signature field is an Acrobat form field. Signature fields are automatically created at the time of signing, but it is also possible to create empty signature fields for later signing.

To create a signature field:

1. Choose **Forms > Add or Edit Fields**.
2. Select the **Add New Field** button.
3. Choose **Digital Signature**.
4. Click and drag where the field should appear. The Digital Signature Properties dialog appears.
5. For simple signature fields, choose **Close**.

   By default, field names are numbered sequentially starting with "Signature1" and contain the default tooltip "Unsigned signature field (click to sign)."

Figure 12 Signature field: Default appearance



6. If you would like to set field properties, choose the **Show All Properties** link in the popup box (for details, see "Specifying General Field Properties" on page 23); otherwise, exit the forms editor.

2.3.2 Specifying General Field Properties

A signature field's general properties include name, tooltip, display behavior, and so on. For example, fields are numbered sequentially and are associated with a generic tooltip. However, the field can be given a unique name, provided with tooltip instructions for an eventual signer, and configured to display only in the Signatures tab and not in the document.

Note: You cannot edit these properties during signing workflows. An author must create a blank signature field and edit the properties before initiating the signing process. Moreover, invisible field properties cannot be edited.

To change a field's general properties.

1. Create a new field.

   Note: For existing fields, place them field in edit mode by selecting **Forms > Add or Edit Fields** and then double click on them OR right click and choose **Properties**.

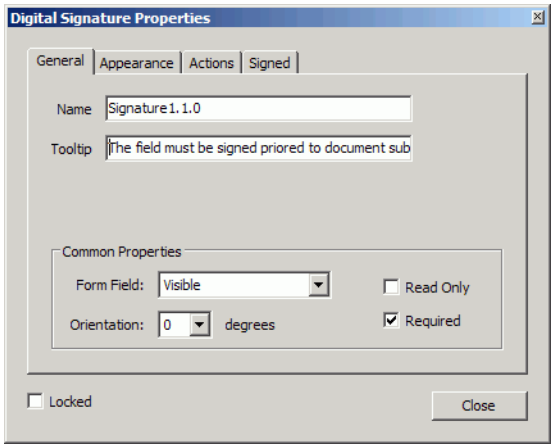
2. Display the General tab.

3. Configure the options:
   - **Name**: Any arbitrary name.
   - **Tooltip**: Any arbitrary text. It is required to make the document accessible to the visually impaired.
   - **Form Field**: Set the field display properties.
     - **Visible**: The field appears in the document, the Signatures tab, and prints.
     - **Hidden**: The field only appears on the Signatures tab and doesn't print.
     - **Visible but doesn't print**: The field appears in the document and Signatures tab but doesn't print.
     - **Hidden but printable**: The field only appears on the Signatures tab and does print.
   - **Orientation**: The field content (signature) orientation AFTER signing. The field does not change.
   - **Read Only**: Prevents changes to the field. Selecting this option prevents signing.

- **Required**: Sets a flag that can be checked by other actions and processes that are dependent on the signature. For details, see "Making a Field a Required Part of a Workflow" on page 29.

4. Edit the properties on the other tabs or choose **Close**.

Figure 13 Signature field: General properties



2.3.3 Customizing Field Appearances

Field border properties, fill color, fonts, and so on can be individually specified. These properties are NOT editable during signing workflows. An author must create a blank signature field and edit the properties before initiating the signing process. Invisible field properties cannot be edited.

To change a signature field's appearance:

1. Create a new field.

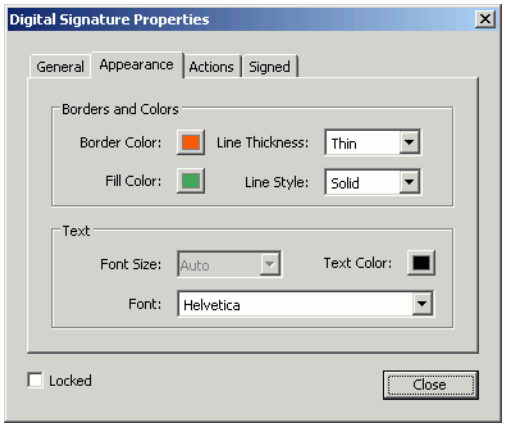
   Note: For existing fields, place them field in edit mode by selecting **Forms > Add or Edit Fields** and then double click on them OR right click and choose **Properties**.

2. Display the Appearance tab.

3. Configure the appearance options.

4. Edit the properties on the other tabs or choose **Close**.

Figure 14 Signature field: Appearance properties



2.3.4 Changing the Default Field Appearance

The default appearance of a blank signature field is a light blue box with no borders that performs no action on signing. These defaults can be changed globally so that all future signature fields will have a custom appearance and action.

To change signature field defaults:

1. Only the attributes on the appearance and Actions tab can be set as defaults for future fields. Customize a field as described in the following.
   - Customizing Field Appearances
   - Specifying a Post-Signing Action
2. Choose **Close**.
3. Right click on the field.
4. Choose **Use Current Properties as New Defaults**.

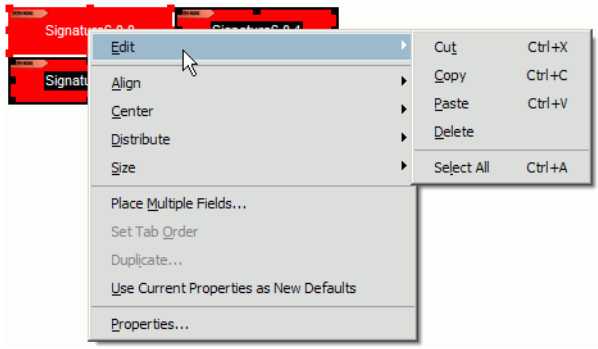
2.3.5 Cut, Copy, and Paste Signature Fields

The forms field context menu provides a number of editing items, including options for cutting, copying, pasting, and deleting.

To perform an edit action on a field:

1. Place the field in edit mode by selecting **Forms > Add or Edit Fields**.
2. Right click on the field.
3. Choose **Edit**.
4. Choose **Cut, Copy, Paste, or Delete** (Figure 15).

Figure 15 Signature field: Edit options



2.3.6 Arranging Signature Fields

While you can drag and drop fields anywhere, the field context menu provides a number of options for arranging multiple fields such as aligning, centering, and distributing fields.

To arrange multiple fields:

1. Place the fields in edit mode by selecting **Forms > Add or Edit Fields**.
2. Drag a rectangle around the fields to arrange.
3. Right click.
4. Choose **Align**, **Center**, or **Distribute** and use the submenus to arrange the fields (Figure 15).

2.3.7 Creating Multiple Copies of a Signature Field

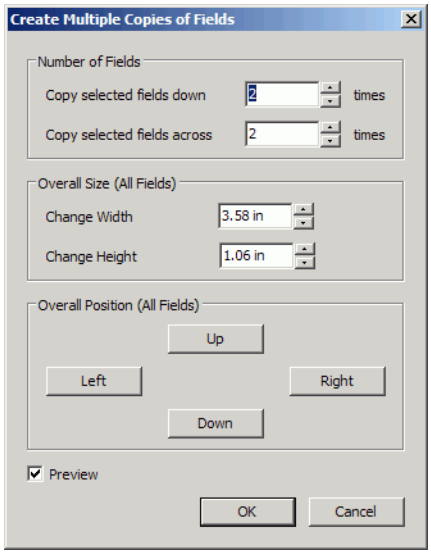
Once a field is configured, multiple copies of the field can be placed on the same page.

To create multiple copies of a field:

1. Place the field in edit mode by selecting **Forms > Add or Edit Fields**.
2. Right click on the field.
3. Choose **Create Multiple Copies** (Figure 15).
4. Configure the copy options, including:
   - The number of fields down and across.
   - The overall field size.
   - The overall position.
5. Choose **OK**.

Tip: Acrobat automatically names the fields by numbering them. Providing unique and intuitive names helps signers and other document recipients navigate and interact with the document.

Figure 16 Signature field: Multiple copy options



2.4 Authoring Signable Forms

Many documents that require signatures are forms. Some forms may have multiple signatures fields, with different signers providing data in certain other form fields. In such cases, document design, field layout, and even field appearance may contribute to the ease with which the form can be integrated into an efficient business process.

For example, it is often useful to lock the form fields associated with a particular signature field once it is signed. This eliminates the need to examine the signed document version to see if the value of a field was changed between that signed version and the current version. For more information, see the following:

- "Authoring a Document with Multiple Fields" on page 27
- "Locking Fields Automatically After Signing" on page 28
- "Making a Field a Required Part of a Workflow" on page 29
- "Specifying a Post-Signing Action" on page 30
- "Unlocking a Field Locked by a Signature" on page 32

2.4.1 Authoring a Document with Multiple Fields

Documents commonly have multiple form fields, and one or more signature fields are often used to verify or approve the data in preceding fields. In these cases, proper document layout and field design may be a critical aspect of usability. When designing a complex form for signing, consider using the following field properties:

- **Layout**: Design the form so that form data precedes a signature. If there is more than one signature field, make sure end users can understand which signature fields are associated with specific data.

- **Appearance**: Signature fields can look similar to other form fields, but it may be desirable to customize their appearance so they can be more readily distinguished. For details, see "Customizing Field Appearances" on page 24
- **Names and tooltips**: Intuitive field names and tooltips facilitate authoring and signing in the following ways:
  - Help the author choose which fields should be read only in the Signed tab of the Digital Signature Properties dialog as well as what field to call when JavaScript is used to customize a document.
  - Help signers find fields and understand how to use the form. For details, see "Specifying General Field Properties" on page 23.
  - Make it easier for signature validators to identify which fields have changed since the names may be used in the Signature pane and elsewhere.
- **Locking behavior**: Consider which fields should become read-only after signing. Locking certain fields helps prevent document changes that could cause a signature to become invalid. For details, see "Locking Fields Automatically After Signing" on page 28.

2.4.2 Locking Fields Automatically After Signing

Form authors can designate which form fields should be locked after any other field is signed. Both signed and unsigned signature fields can be configured to become read-only after they are signed. By setting post-signing, field locking properties, authors can prevent data changes to any combination of form or signature fields. Two common use cases for automatic locking include:

- Preventing users other than the document author from clearing or re-signing a field.
- Preventing users from changing form data after the document has been signed.

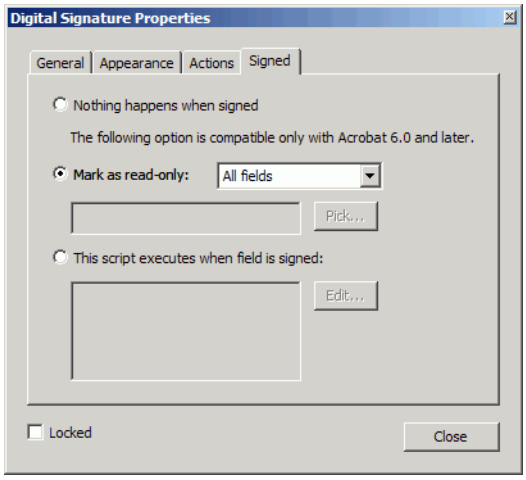
To automatically lock one or more fields after signing:

1. Create a new signature field.

   Note: For existing fields, place them field in edit mode by selecting **Forms > Add or Edit Fields** and then double click on them OR right click and choose **Properties**.

2. Display the Signed tab.

Figure 17 Signature field: Signing properties



3. Choose **Mark as read only**.

   When a field with field locking is signed, both a normal document signature and an object hash of the locked fields are produced and included in the document. When the signature is validated, the viewing application validates the bytes of the PDF file AND compares the object hash in the signature to the object hash from the objects in memory. This allows the application to detect prohibited, changes.

4. Use the drop-down list to select from the following:
   - **All fields**: All signature fields will be read only after signing.
   - **All fields except these**: All signature fields except those specified will be read only after signing. Choose **Pick** and select the field to exclude.
   - **Just these fields**: Only the specified signature fields will be read only after signing. Choose **Pick** and select the field to include.

5. Choose **Close**.

2.4.3 Making a Field a Required Part of a Workflow

Certain workflows may require a signature. For example, after a signature field is signed, form fields may be prepopulated or additional fields may appear. It is also common for form designers to require signing before the document can be emailed or submitted to a server for processing.

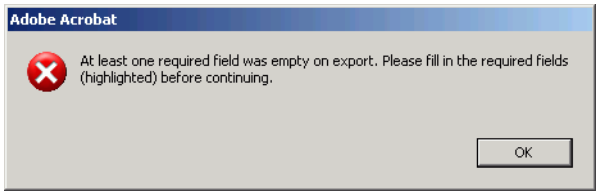
To require a signature:

1. Place the field in edit mode by selecting **Forms > Add or Edit Fields**.
2. Right click on the field and choose **Properties**.
3. Check **Required** on the General tab.
4. Choose **Close**.

Users can still open, close, save, and send the document without any indication that the field is required until the document author sets up a check for the required flag. For example, the check could be as

simple as emailing the document. In this case, the author would add an action to the button to submit the document and configure a URL. If the document recipient clicks on the field and then cancels the signing process, an alert will appear. Server-side and other JavaScript checks are also possible.

Figure 18 Required field not signed alert



2.4.4 Specifying a Post-Signing Action

JavaScript actions can be associated with a signature field so that an action occurs whenever the user interacts with the field in some predefined way. However, documents are usually signed to protect, guarantee, and or attest to the signed content. Signers usually want to know that the document they are seeing is the document they are signing, and document recipients usually need to know that the document they are viewing is the same as the document that was signed. For this reason, adding actions to a signature field is inadvisable. Field actions change the underlying bytes of a PDF and could adversely affect document security as well as content integrity.

Caution: Using this feature is NOT recommended since such actions are contrary to the secure and trusted nature of most signing workflows. Adding actions will result in a legal warning about the legal integrity of the document.

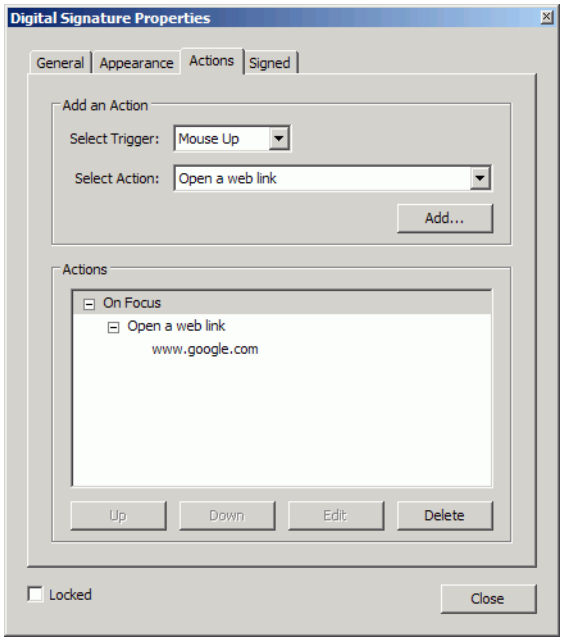
To associate an action with a field.

1. Create a new field.

   Note: For existing fields, place the form in edit mode by selecting **Forms > Add or Edit Fields** and then double click on them OR right click and choose **Properties**.

2. Display the Action tab.

Figure 19 Signature field: Action properties



3. Configure the options:
   - **Select Trigger**: Choose a type of action.
     - **Mouse Up**: The user clicks on the field and releases.
     - **Mouse Down**: The user clicks on the field.
     - **Mouse Enter**: The cursor enters the field.
     - **Mouse Exit**: The cursor exits the field.
     - **On Focus**: The user hovers over or tabs to the field.
     - **On Blur**: The user stops hovering over or tabs away from the field.
   - **Select Action**: See Table 5.
4. Choose **Add**.
5. Follow the action instructions that appear in the action dialog.
6. Optional: Move actions **Up**, **Down**, **Edit**, or **Delete** actions as necessary.
7. Choose **Close**.

Table 5 Actions that can be associated with a signature field

Action	Description
Execute a Menu Item	Executes a specified menu command as the action.
Go to 3D View	Changed the view to the 3D view specified by the form author.
Go to a Page View	Jumps to the specified destination in the current document or in another document.

Table 5 Actions that can be associated with a signature field

Action	Description
Import Form Data	Brings in form data from another file, and places it in the active form.
Open a File	Launches and opens a file. If you are distributing a PDF file with a link to a non-PDF file, the reader needs the native application of the non-PDF file to open it successfully. (You may need to add opening preferences for the target file.)
Open a Web Link	Jumps to the specified destination on the Internet. You can use http, ftp, and mailto protocols to define your link.
Play a Sound	Plays the specified sound file. The sound is embedded into the PDF document in a cross-platform format that plays in Windows and Macintosh.
Play Media (Acrobat 5 Compatible)	Plays the specified QuickTime or AVI movie that was created as Acrobat 5-compatible. There must already be a link to the movie in the PDF document for you to be able to select it. (See Adding movie clips.)
Play Media (Acrobat 6 Compatible)	Plays a specified movie that was created as Acrobat 6-compatible. There must already be a link to the movie in the PDF document for you to be able to select it. (See Adding movie clips.)
Read an Article	Follows an article thread in the active document or in another PDF document.
Reset a Form	Clears previously entered data in a form. You can control the fields that are reset with the Select Fields dialog box.
Run a JavaScript	Runs the specified JavaScript.
Set Layer Visibility	Determines which layer settings are active. Before you add this action, specify the appropriate layer settings.
Show/Hide a Field	Toggles between showing and hiding a field. This option is especially useful in form fields. For example, if you want an object to pop up whenever the pointer is over a button, you can set an action that shows a field on the Mouse Enter trigger and hides a field on Mouse Exit.
Submit a Form	Sends the form data to the specified URL.

2.4.5 Unlocking a Field Locked by a Signature

When a signature field's properties specify that signing will automatically lock other fields, those fields cannot be edited until they are unlocked. Since it was a signature that locked the fields in the first place, unlocking the fields simply involves clearing the signature.

Tip: Unlocking a field is the same thing as clearing the signature field. Only the signer clear the signed field.

To unlock a field:

1. Right click on the signature field.
2. Choose **Clear Signature**.

   Other fields can now be edited.

3 Validating Signatures

When you receive a signed document, you may want to validate its signature(s) in order to verify who the signer was, when they signed it, and what was actually signed. Depending on how you have configured your application, validation may occur automatically.

However, understanding both how to validate a signature manually as well as what signature components are analyzed during the validation process can facilitate trouble-free workflows and mitigate signature status problems. Participants in signing workflows may also want to configure their environment to streamline the validation process and control what kinds of content in signed documents can be run on their machine.

The following sections provide validation details:

3.1 Signature Validity Basics

As part of the signature validation process, Acrobat and Adobe Reader verify the signer's identity as well as the document's integrity.

3.1.1 What Makes a Signature Valid?

Signature validity is determined by checking the signer's digital ID certificate status (*is it valid and trusted?*) and document integrity (*has it changed since being signed?*):

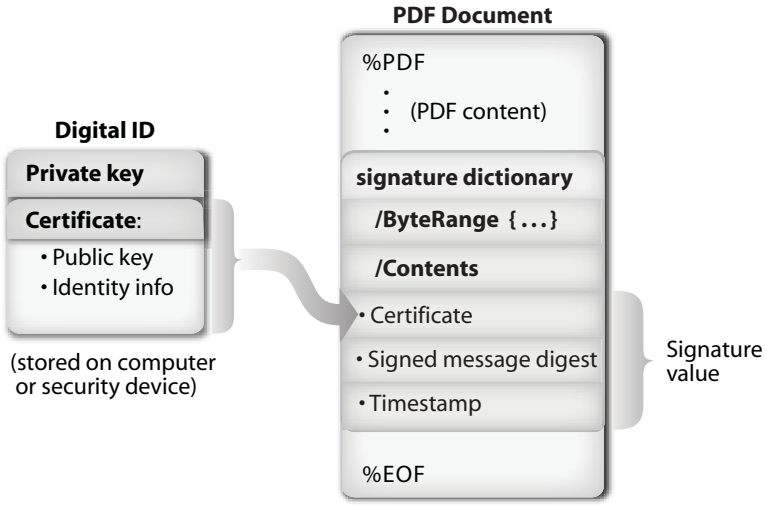
- Authenticity verification confirms that the signer's certificate or that one of its parent certificates exists in the validator's list of trusted identities and that the signing certificate is valid at that point in time according to the user's Acrobat or Reader configuration (time of signing, secure timestamp time, or current time).
- Document integrity verification confirms that the signed content hasn't changed after it was signed or that it has only changed in ways specifically permitted by the signer.

3.1.1.1 Authenticity Verification

Authenticity verification starts with a signer obtaining a digital ID that includes an X.509 certificate. The validator must add that certificate (or have previously added one of its issuing certificates) to their trusted identities list. Either the signer's certificate or one of its issuing certificates must then be explicitly trusted for signing, thereby making it a trust anchor used during signature validation. At

validation time, the certificate is processed and analyzed to see if it's valid. That is, Acrobat performs a revocation check and other relevant operations before determining what the signature status will be.

Figure 20 Internal Document Signature components



3.1.1.2 Document Integrity Verification

In signing workflows, document integrity refers to whether or not what was signed has changed after signing in a way that violates any document rules. That is, what the signer signed should be reproducible and viewable on the document recipient's end. At a high level, the Acrobat family of products therefore implements signatures as follows:

- Each signature captures what the document looked like at the signing point in time.
- Only very limited changes are possible after a signature is applied. At most, form field values, additional signatures, and annotations can be changed or added.
- View Signed Version shows exactly what was signed. The signature panel lists post-signing changes.
- A certification signature can tighten the rule to allow less changes than form fields, additional signatures, and annotations.

To verify if a document has changed after signing (has integrity), Acrobat or Adobe Reader must have a way to uniquely identify what was signed. To do this, it uses a *message digest*. A message digest is a number which is created algorithmically from a file and which uniquely represents that file. If the file changes, the message digest changes. Sometimes referred to as a *checksum* or *hash*, a message digest is simply a unique number created at signing time that identifies what was signed and is then embedded in the signature and the document for later verification.

During the act of signing, the application creates a message digest and then encrypts that digest with the signer's private key. The digest is embedded in the document along with the signature appearance. Every time a document is signed, a new digest is created. Thus, each signature is only valid for a specific version of the document.

Because the application stores and numbers a document version for each signature, signature validators can determine what was actually signed. When you validate a signature, a new message digest is created and compared to the digest that was embedded in the document at signing time. If the two digests are not identical the signature is invalid.

Both signers and signature validators should understand the following about the relationship between signatures and document versions:

- Every time a document is signed, the document's state at the point of signing is stored in the PDF.
- Versions are incrementally numbered beginning with "1."
- A document with 10 signatures will have 10 versions.
- A signature applies to a version (e.g. signature X with version X and signature Y with version Y, etc.).
- When you open a document in Acrobat or Adobe Reader, the current version always displays.

Note: To learn more about how each signature results in a new version of the document, refer to http://www.adobe.com/devnet/acrobat/pdfs/DigitalSignaturesInPDF.pdf.

3.2 Setting up Your Environment for Signature Validation

Document recipients should configure their environment to handle incoming documents in a way that enhances workflow efficiency or meets some business need. While Adobe Acrobat and Adobe Reader provide default options, customizing the environment often provides a better user experience. In large, enterprise environments, your environment may be preconfigured by a system administrator.

Options include the following:

- Validating Signatures Automatically: By default, validation occurs automatically. If signatures should not be validated automatically when a document opens, turn this option off.
- Validation Method, Revocation Checking, and Time Preferences: Accept the defaults or configure plugin usage, time display, automatic revocation checking, and other settings.
- Using Root Certificates in the Windows Certificate Store: If you would like to trust and use certificates in the Windows Certificate Store for signature validation, turn this option on. Trusting all of these certificates is not recommended.
- Certificate Trust Settings: Specify whether a certificate should be a trust anchor, trusted for signing, and trusted for certain behaviors in certified documents.

3.2.1 Validating Signatures Automatically

By default, signatures are automatically validated. However, you may want to turn it off because:

- You don't care whether the signatures are valid.
- The desktop cannot be configured to validate the signature.
- To gain a small increase in application speed when it opens a document. The difference may be negligible depending on the number of signatures and whether a system administrator has customized revocation checking.

To configure automatic signature validation:

1. Choose **Edit > Preferences** (Windows) or **Acrobat (or Adobe Reader) > Preferences** (Macintosh).
2. Choose **Security** in the left-hand list.
3. Check **Verify signatures when document is opened**.

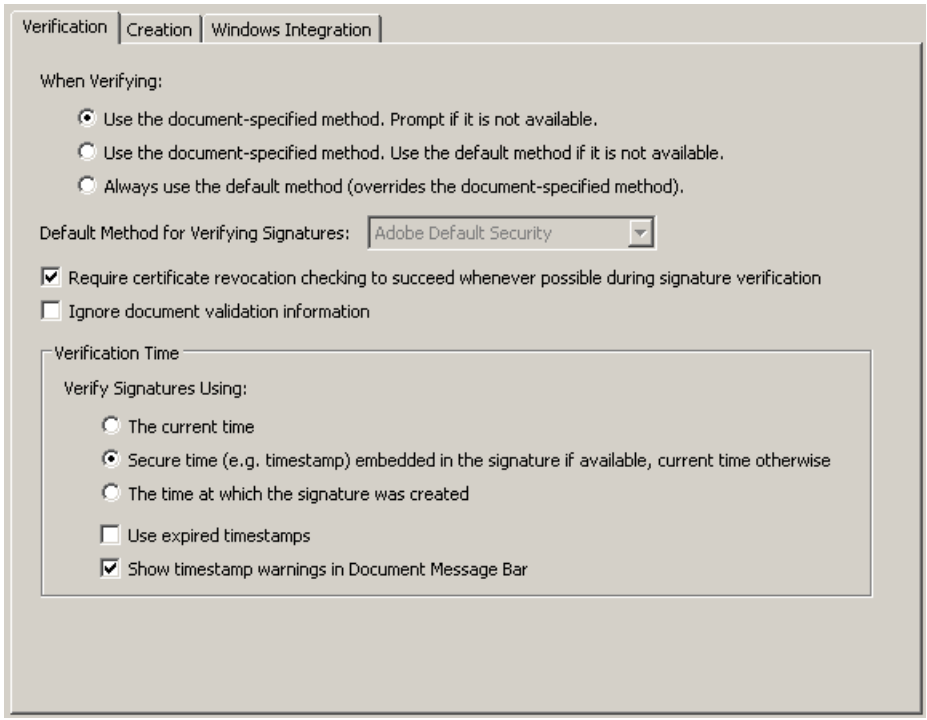
3.2.2 Validation Method, Revocation Checking, and Time Preferences

To set advanced digital signature preferences:

1. Choose **Advanced Preferences**.
2. Display the Verification tab.

   Verification tab options let you specify the validation plugin, default validation methods, whether or not certificate revocation checking is automatic, what time is associated with a validated signature, and whether or not a status icon appears with the signature.

Figure 21 Signature verification preferences



3. Select the signature validation method (use the default setting unless instructed to change it by a system administrator):
   - **Use the document-specified method, prompt if it is not available**.
   - **Use the document-specified method, use the default method if it is not available**.
   - **Always use the default method (overrides the document-specified method)**.

   Tip: Don't change this setting unless instructed to do so by a system administrator. Signatures are created and validated by plugins. These options specify which plugin is used. Both Acrobat and Adobe Reader provide a default plugin for signing documents

and verifying signatures. While the signing and verification usually use the same plugin, this is not always the case. However, a signature always "knows" what plugin is required to verify it.

4. If you have installed a non-default plugin, select your preferred method for verifying signatures.

5. Check or uncheck **Require that certificate revocation checking be done whenever possible during signature validation**.

   This option checks certificates against a list of revoked certificates during validation, either with the Online Certificate Status Protocol (OCSP) or the Certificate Revocation List (CRL). If this option is not selected, the revocation status for approval signatures is ignored. *Revocation checking always occurs for certificates associated with certification signatures.*

   Note: Signature verification is similar to credit card validation. OCSP checking is like making a phone call to verify the card number. CRL checking is like checking the card numbers against a list.

6. In the Verification Time panel, select a time verification method:
   - **Current time**: The digital signature validation time.
   - **Secure time**: The secure timestamp server time if one is present and trusted, otherwise the current time.
   - **Creation time**: The signature creation time. This may be synonymous with the secure time if a timestamp was used.

Time verification changes from 9.0 to 9.1

With 9.1, the default preferences assure signatures will be valid if the certificate used was valid at the time of signing rather than valid at the time of checking the validation. So by default, a signature will remain valid in the long term even after the certificate has expired. But it also means that, as well as trusting the signature, you are also trusting the document's signing time.

Tip: It is possible for a signer to change their system date to a time when a certificate was valid and then sign the document, which would then lead to misleading results. Hence, it is suggested that signatures be configured to use time stamp servers.

In Acrobat 9.0, if the verification time preference is set to "The time at which the signature was created", then the CRL validity window had to include the time-of-signing in order to be used in signature validation. In Acrobat 9.1, this behavior is changed so that the CRL validity window overlaps the interval from the time-of-signing to the end of validity of the signing certificate. This change allows CRLs issued after the time-of-signing to be more consistently used. This change can result in a behavior change where signatures that could not be validated (but were in the problematic state) can now be successfully validated, yielding a valid or invalid signature status.

3.2.3 Using Root Certificates in the Windows Certificate Store

The Windows Certificate Store contains a store called "Trusted Root Certificate Authorities" that contains numerous root certificates issued by different certification authorities. Certificates are "root" certificates by virtue of being at the top of the certificate chain hierarchy. There are two common ways a certificate ends up in the Windows Certificate Store root directory:

- The computer manufacturer or Microsoft has put them there.

- A company administrator has put them there as part of a company-wide program.

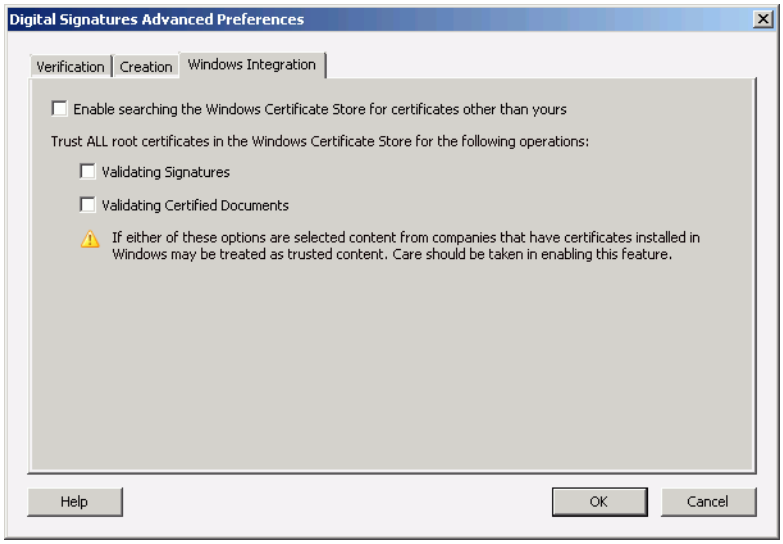
Most home users should not trust all Windows root certificates by default because by trusting a root certificate you may be trusting all the content provided by the company that owns that certificate. Many root certificates ship with Windows, and users may have imported others as a result of some online action.

Enterprise users, on the other hand, should consult company policy to determine whether or not to trust all Windows root certificates for validating signatures or certifying documents. This information should come from an administrator, though your application may already be configured with the correct settings. A common reason to trust Windows roots is so the administrator can manage from a central location the certificates deployed on a network.

To use these certificates for signature validation:

1. Display the Windows Integration tab.

Figure 22 Trusting Windows root certificates



2. Specify the trust level for all root certificates in the Windows Certificates Store:
   - **Validating signatures**: Certificates will be trusted for approval signature validation.
   - **Validating certified documents**: Certificates will be trusted for certification signature validation.
3. Choose **OK,** and exit the preferences dialogs.

3.2.4 Validating Signatures with Timestamps and Certificate Policies

Certificate policies can be used with timestamps, but they can only be verified on the client end, not on the server end. That is, a timestamped signature can not be sent with CRL request with a specific policy OID; however, the client can require that the server response include a specified policy constraint. If the timestamp server returns a response that doesn't include a matching policy OID, then the client would reject the timestamp and it's status would be invalid. The user interface shows the following:

- The signature could be valid, but it's validated at the current time.

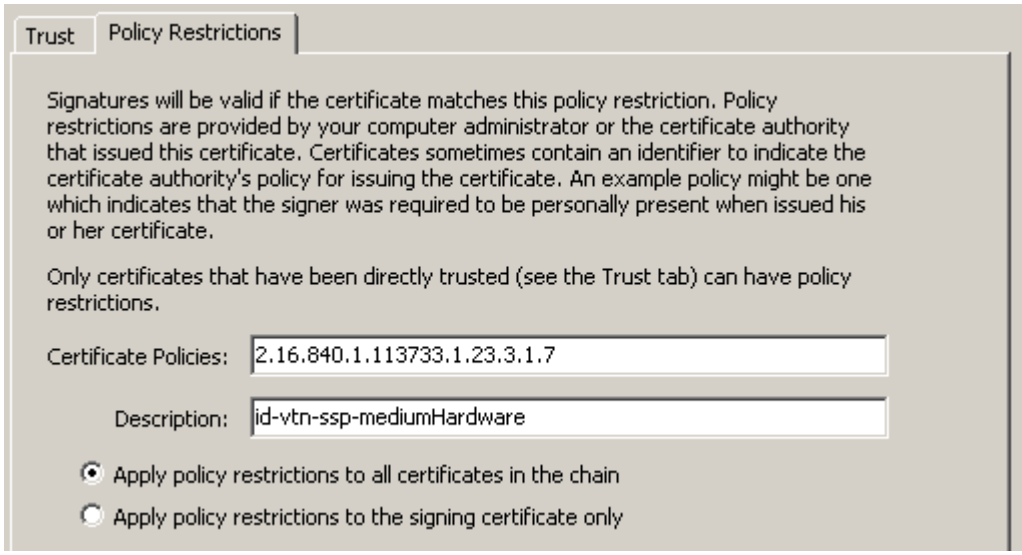
- The timestamp is invalid. The Summary tab of the Signature Properties dialog shows a red X.

To require a timestamp to be associated with a particular certificate policy:

1. Configure your application to validate signatures using Secure Time as described in "Validation Method, Revocation Checking, and Time Preferences" on page 36.
2. Configure a policy constraint for a trust anchor in your trusted identities list:
    1. Choose **Managed Trusted Identities** (See Content security menus).
    2. In the **Display** drop down list, choose Certificates.
    3. Select the timestamp server's certificate that will be used as a trust anchor.
    4. Choose **Edit Trust**.
    5. Choose the Policy Restrictions tab and enter the restrictions:
        - **Certificate Policies**: Required. Enter the policy OID.
        - **Description**: Optional. Enter a meaningful description.
    6. Choose the radio button that specifies whether to apply policy restrictions to all certificates in the chain or only to the signing certificate. The latter choice is supported by 10.0 and later only.
    7. Choose **OK**.

        Note: If the timestamp server returns a response with a policy not specified by the client, the timestamp signature will be invalid due to an invalid policy constraint.

Figure 23 Policy restrictions



3.3 Validating Signatures Manually

Unless the application is configured to do otherwise, signatures are validated automatically when a document opens. You can always validate one or more signatures manually.

Validating a signature allows you to verify the signer's identity and determine whether the displayed document is identical to what was signed (or that only allowed changes were made):

- Identity verification confirms the signer's certificate or one of its parent certificates exists in the list of trusted identities and is not expired or revoked.
- Document integrity verification confirms that the signed content hasn't changed since signing or that it has only changed in ways specifically permitted by the signer.

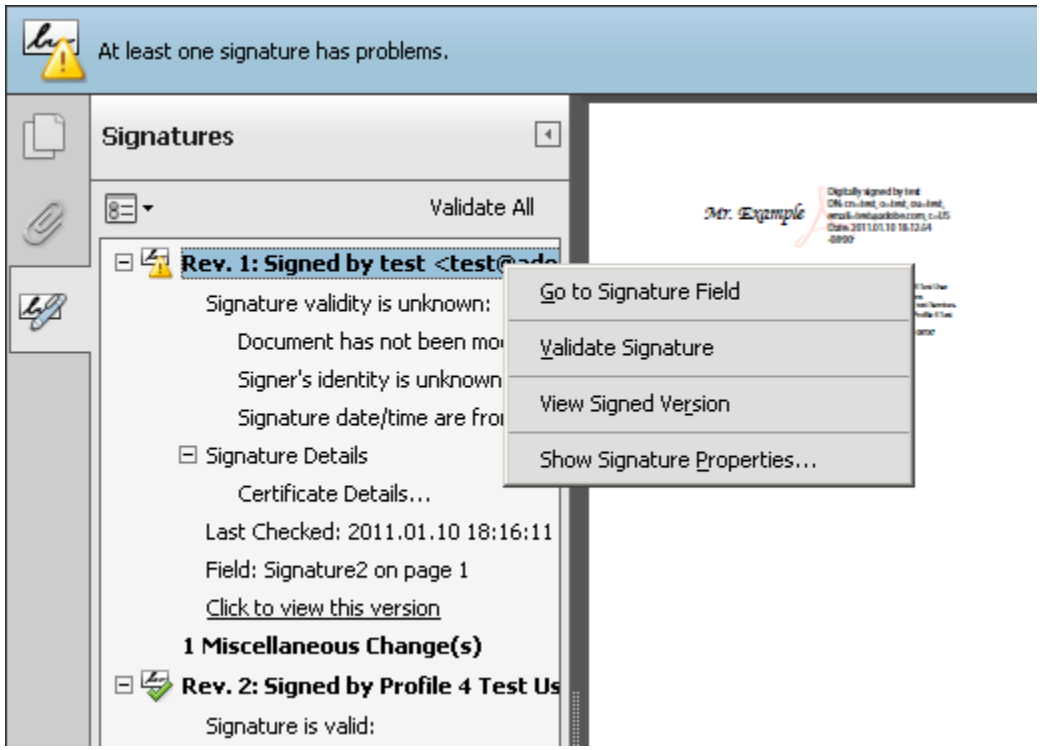
Before validating a signature, it is a good idea to understand what a signature is and how signature status is indicated. For details, see "What Makes a Signature Valid?" on page 33 and "Status Icons and Their Meaning" on page 46.

3.3.1 Validating Signatures

All signatures in a document may be validated one at a time or simultaneously. This feature is particularly useful if the auto-validate option has been turned off.

- Verify a single signature manually by right clicking on any signature in the Signatures pane or in the document and choosing **Validate Signature**.
- Validate all signatures by choosing **Validate All** in the Signatures Pane.

Figure 24 Signatures tab: Validate signature



3.3.2 Validating a Problematic Signature (trusting a signer on-the-fly)

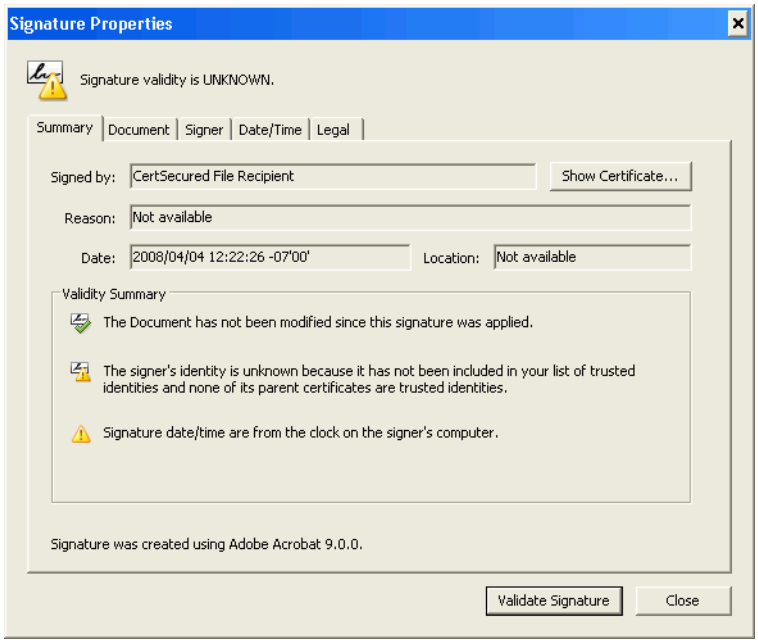
If a signer's digital ID certificate has not been explicitly trusted, the signer is untrusted and the signer's signature validity will be *problematic*. When a signer has not been trusted ahead of time, you can trust their certificate for signing and certifying directly from the signature. After their ID (contact information and certificate) is added to your list of trusted identities, the signature can be validated.

To add an someone's certificate a list of trusted identities:

1. Display the Signature Properties dialog by right clicking on any signature in the document or the Signatures tab and choosing **Show Signature Properties**.

2. Choose the Summary tab (Figure 25).

Figure 25 Signature Properties: Summary



3. Choose **Show Certificate**.

   Adding an unverified digital ID certificate to the trusted identity list could pose a security threat. This is particularly true for self-signed IDs that are not issued by a third-party certificate authority. For details, see "Verifying the Identity of Self-Signed Certificates" on page 51.

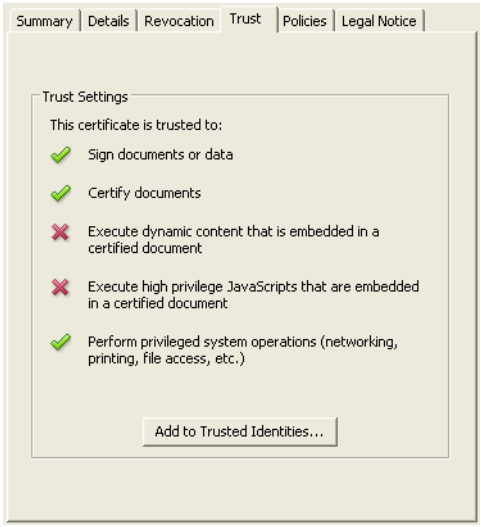
4. When the Certificate Viewer appears, choose the Trust tab (Figure 26).

5. Choose an item in the left-hand certificate path field. There may be one or more certificates which make up a certificate chain.

   Tip: If the bottom-most certificate on the chain is selected, then only that certificate will be trusted. If the top-most certificate is selected, then any certificates having that certificate as a root will be trusted. For example, if the root certificate is issued by VeriSign and it is trusted, then other certificates having VeriSign's certificate as the root (also issued by them) will also be trusted. It is a best practice to trust the topmost certificate that you are willing to trust. Revocation checking starts at the bottom of a

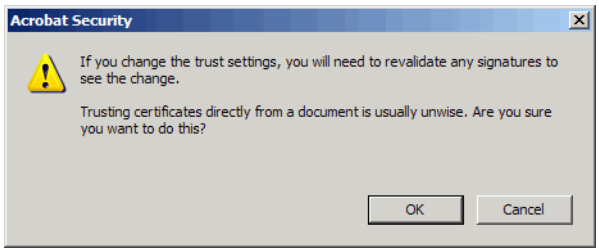
chain (begins with the end entity), and once it reaches a trusted root revocation checking stops.

Figure 26 Certificate viewer: Trust tab



6. Choose **Add to Trusted Identities**.
7. When asked if the certificate should be trusted, choose **OK**.

Figure 27 Trusting certificate from a document warning



8. When the Import Contact Settings dialog appears, configure its trust levels.

   The Policy Restrictions tab will not appear if there are no policies associated with this certificate.
9. Choose **OK**.
10. Choose **OK**.
11. Choose **Close**.
12. Right click on the signature and choose **Validate Signature**.

    Tip: The yellow triangle icon on the signature will not change until the signature is revalidated.

3.3.3 Validating Signatures for other Document Versions

When you open a document, the latest version is always displayed. You can see whether the signature associated with earlier signed versions of the document are valid simply by opening the Signature pane and viewing the status icon and text.

Documents with multiple signatures contain the elements needed to reconstruct any previous version of itself as it existed at the time of signing. In other words, Acrobat and Adobe Reader "remembers" that version A is signed, that changes were made to version B, and so on. Therefore, it may be necessary to view the signed version in order to see what content was actually signed. Viewing the signed version allows you to check if the signature is valid for a particular document version.

To view the signed version of a document.

1. Right click on the signature you want to validate in the document or in the Signatures tab.
2. Choose **View Signed Version**. The application opens the signed version of the document.
3. Revalidate the signature if necessary.

Tip: For more about versioning, see "Document Integrity Verification" on page 34.

3.3.4 Validating Signature Timestamps

If you know a signature is timestamped or your workflow requires timestamps, read the following sections. At a high level, the rules are as follows:

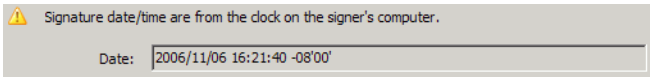
- You can configure Acrobat to use timestamps by setting the verification preferences as described in "Validation Method, Revocation Checking, and Time Preferences" on page 36.
- If set, the secure timestamp server time is used if one is present and trusted, otherwise the current time is used.
- Timestamp validity does not affect signature validity. A signature can be valid even if the timestamp server's certificate is invalid or expired.
- The signature validation time appears in the Date/Time tab of the Signature Properties dialog.

Local Time versus Timestamp Time

Signature times tell you that a document and signature existed prior to the indicated time. All signatures are associated with the signer machine's local time, but they may also include a timestamp time if the signer's application was configured to use a timestamp server. Because users can set their machine time forward or back, local time is less reliable than a timestamp time. Local times are labelled as such in the Date/Time and Summary tabs of the Signature Property dialog (Figure 28).

Note: Because signature appearances only display local time, the appearance time will be different from the timestamp time shown in the Date/Time tab of the Signature Properties dialog.

Figure 28 Timestamps: Local, machine time



What is a timestamp?

A timestamp is like a signature inside of a signature. Like signatures, timestamps are provided by someone (a timestamp authority) who uses a certificate to confirm their identity. A timestamp's certificate must be valid (not revoked by the issuer) and trusted (by you) for the timestamp to be valid. Timestamp certificate status appears in the Date/Time and Summary tabs of the Signature Property dialog:

- Untrusted timestamp certificates appear as follows:

Figure 29 Timestamps: Untrusted stamp

- Trusted timestamps that have been added to the Trusted Identities list and have been explicitly trusted for signing appear as follows:

Figure 30 Timestamps: Trusted stamp

How Do I Validate a Timestamp in a Signature?

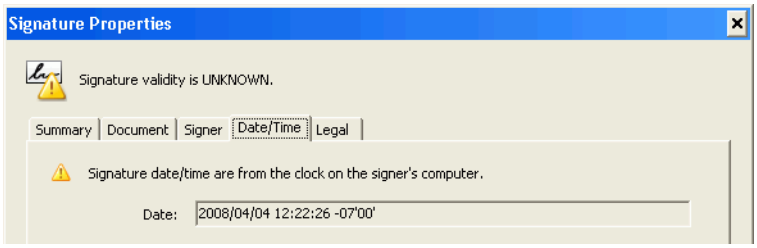
Validating a timestamp is the process by which you check to see if the timestamp was applied and that its certificate is valid. In order to validate a timestamp, you need to manually verify:

- **The timestamp was applied**: If a timestamp fails for some reason (the server cannot be found, the network is down, etc.), timestamping fails silently.
- **The timestamp certificate is trusted**: If a timestamp was applied, the certificate must be trusted by adding to your trusted identities list.

To verify that a signature has been properly timestamped:

1. Display the Signature Properties dialog by right clicking on any signature in the document or the Signatures tab and choosing **Show Signature Properties**.
2. Choose the Date/Time tab. Timestamp status is indicated by the icon and associated text:
   - **Warning triangle**: Timestamping failed or a timestamp is not present and the local time is used. Verify the signer used a timestamp.
   - **Magnifying glass**: A timestamp may have been used but you have not yet trusted the timestamp certificate. Proceed to the next step.
   - **Clock**: A timestamp with a trusted certificate was used.

Figure 31 Timestamps: Date/Time tab



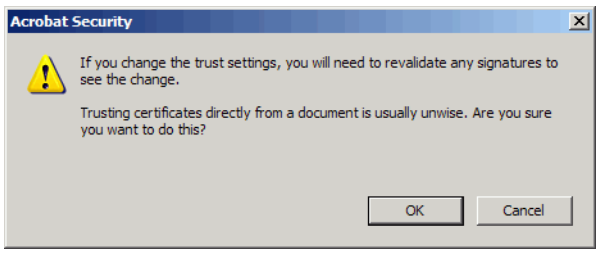
Note: The following steps add a timestamp certificate to your list of trusted identities.

3. Choose **Show Certificate**.
4. When the Certificate Viewer appears, choose the Trust tab.
5. Choose an item in the left-hand certificate path field. There may be one or more certificates which make up a certificate chain.

   Tip: If the bottom-most certificate on the chain is selected, then only that certificate will be trusted. If the top-most certificate is selected, then any certificates having that certificate as a root will be trusted. For example, if the root certificate is issued by VeriSign and it is trusted, then other certificates having VeriSign's certificate as the root (also issued by them) will also be trusted. It is a best practice to trust the topmost certificate that you are willing to trust. Revocation checking starts at the bottom of a chain (begins with the end entity), and once it reaches a trusted root revocation checking stops.

6. Choose **Add to Trusted Identities**.
7. When asked if the certificate should be trusted from within the document, choose **OK**.

Figure 32 Revalidate signatures warning



8. When the Import Contact Settings dialog appears, configure the its trust levels.

   The Policy Restrictions tab will not appear if there are no policies associated with this certificate.

9. Choose **OK**.
10. Choose **OK**.
11. Choose **Validate Signature** on the Date/Time tab of the Signature Properties dialog. The icon should change to a clock if the check is successful.

3.3.4.1 When Timestamps Can't be Verified. . .

If a signature is timestamped but cannot be verified, open the Trusted Identity Manager and verify:

- A certificate is associated with the timestamp server. Verify the timestamp authority's certificate is in the certificate list.
- The trust level of the certificate is set. Choose a certificate and verify that the trust level is set for signing. The certificate must either be a trust anchor or be issued by someone whose certificate you have specified as a trust anchor.

3.4 Status Icons and Their Meaning

By default, signatures are validated automatically when a document opens. You can change this behavior as described in "Validating Signatures Automatically" on page 35. Signature and document status' are represented by status icons and text both in the document, on the Signatures pane, on the Document Message Bar, and in the Signature Properties dialog, and elsewhere.

Note: For a higher level of assurance, do not rely solely upon the visual inspection of status icons. Review the Signature Properties dialog for revocation and trust information as well as the signer's certificate details.

3.4.1 Signature Status Definitions

To determine a signature's status, the application checks the signature's digital ID certificate status (is it valid) and document integrity (has it changed since being signed).

The rules for determining signature status are as follows:

- **Valid** signatures used a valid and trusted certificate and the document has not changed or has changed in ways specifically permitted by the author.
- **Problematic** signatures are associated with certificates that cannot be validated or lack a trust relationship with the signer.
- **Unknown** signatures indicate that the signature validity state has not been checked.
- **Invalid** signatures either have an invalid certificate or the document has changed in ways specifically prohibited by the author.

3.4.2 Document Status Definitions

Tip: For a one page key, see "Signature Status and Troubleshooting Guide" on page 103.

In addition to the individual status for each signature, the Document Message Bar displays the document's overall status. The document status is essentially a summation or "rollup" of all the signature status' AND the effect of document changes after the last signature.

For example, a form might have two valid signatures and a valid document status. However, when some someone types into the form's text box, both the signature status' get the blue "i" information icon indicating that something has changed after the signatures were added. The document message bar

now shows a yellow triangle indicating that there are unsigned changed in the document. If the form is signed again, the overall document status changes back to valid as indicated by the green check.

The rules for determining a document's status are as follows:

- If there is only one signature and the document hasn't changed since it was signed, then the document status is identical to the signature status.
- The status is flagged as problematic if there are unsigned changes following the last approval signature.
- The status is unknown (magnifying glass) if the authenticity verification check could not complete.
- Like a signature status, if either the authenticity verification or document integrity check fails, the overall document status is invalid (red x).

3.5 Troubleshooting a Signature or Document Status

Note: In enterprise settings, a system administrator may have configured your application to behave differently than described below.

Ideally, signature validation should result in the display of a green check icon for approval signatures or a blue ribbon icon for certification signatures. Icons always appear in the Document Message Bar, the Signature Pane, and in the Signature Properties dialog.

In addition to the signature and document status icons and text, key tools for troubleshooting signatures include the following:

- **Signatures pane**: Displays all of the signatures, status, change history, and links to signed versions.
- **Signature Properties dialog**: The dialog provides five tabs that display signature information and buttons for performing document validation tasks. It also provides a **Show Certificate** button for invoking the Certificate Viewer.
- **Certificate Viewer**: The viewer provides certificate-specific information and buttons for performing certificate validation tasks. Together, the Signature Properties dialog and Certificate Viewer should provide you with enough information to either successfully validate a signature or reject the document as insecure.

3.5.1 Troubleshooting an Identity Problem

If the signature status or overall document status indicates that there is a problem with verifying the authenticity of the signer, you may need to verify the signer and/or decide whether to trust that signer.

Note: Trust does not happen automatically. For a signature to be trusted, your application must be configured for that trust. That configuration could be the result of actions by Adobe, your administrator, or you.

To troubleshoot authenticity problems, open the signature panel and expand the information for the problematic signature. Read what it says the problem is, and then take one or more of the following actions:

1. If the status is unknown (displays a magnifying glass) and the icon shows a magnifying glass, it is possible signature validation did not occur.
   - Validate the signature(s) as described in "Validating Signatures Manually" on page 40.
   - Verify you have an internet connection and the application is configured properly.
   - Since the problem may not be with your application, try again later.
2. If the status is problematic (displays a warning triangle), do the following:
   - Verify the signer is in your trusted identity list and that you have configured their certificate or one of their certificate's issuing certificates as a trust anchor.
   - Verify you have trusted the signer's certificate for signing and (if necessary) certifying. The Certificate Viewer's Trust tab indicates the certificates trust level (right click on a Signature, choose **Show Signature Properties** and then **Show Certificate**).
   - Verify that a revocation check occurred. Open the Certificate Viewer's Revocation tab (right click on a Signature, choose **Show Signature Properties** and then **Show Certificate**). Check the following:
     - If revocation checking occurred, **Problems encountered** is active and you can select the button to view a description of the problems.
     - If revocation checking did not occur at all, **Check revocation** is active and you can select the button to check revocation manually.
     - If online revocation checking is required, it may have failed as a result of no online access or an application problem.
3. If the status is invalid (displays a red X), the signer's certificate is invalid. Do the following:
   - Contact the signer. The signer may need to get a new digital ID and re-sign a new document.
   - Policy restrictions on a trust anchor can result in signature invalidity. If you have set a policy restriction, determine if that is the problem remove the restriction.
4. If you still cannot pinpoint the problem, or you need help with some of the steps above, read the following:
   - "Troubleshooting Digital ID Certificates" on page 48
   - "Displaying the Signer's Certificate" on page 49
   - "Verifying the Identity of Self-Signed Certificates" on page 51
   - "Checking Certificate Revocation Status" on page 52
   - "Exporting a Certificate Other than Yours to a File" on page 52

3.5.1.1 Troubleshooting Digital ID Certificates

Someone becomes your trusted identity when you import their valid digital ID certificate and set a specific trust level for that certificate. You can set trust levels ahead of time if you have access to those certificates. If you do not have access to those certificates, simply validate and trust certificates "on-the-fly" as you receive individual documents. As shown in Table 8, the Certificate Viewer provides six tabs with functionality for working with and verifying digital ID certificates.

Table 8 Certificate Viewer information

Tab	What it shows	What you can do
Summary	Signer and Issuer information, validity dates, and intended usage.	Export the certificate to a file.
Details	Certificate data such as subject, issuer, used algorithms, public key, and so on.	The data can be used in a variety of ways such as using the digests to verify the certificate's origin.
Revocation	Shows certificate validity status of a revocation check and provides an explanation.	**Signer Details**: Open the certificate in the Certificate Viewer. The butto only active if the revocation check was successfully completed. **Problems encountered**: View revocation checking problems. The but is only active if revocation checking occurred but failed. **Check revocation**: Enables manual revocation checking. The button is only active if no checking occurred AND a check is possible.
Trust	Lists the user-specified certificate trust settings.	Add the certificate to the Trusted Identity list.
Policies	Lists policy OIDs associated with this certificate, if any. Describes the policy.	View policy details.
Legal Notice	Displays a generic legal disclaimer, the certificate issuer's policy statement, issuer notice, and link to the policy, if any.	If an issuer policy is used, the policy can be displayed.

3.5.1.2 Displaying the Signer's Certificate

When a certificate is displayed in the certificate viewer, you can check certificate validity, trust settings, associated policies, and other details that help you establish the owner's identity. The Certificate Viewer provides six tabs that displays certificate data and allows you to manage that certificate (Table 8).

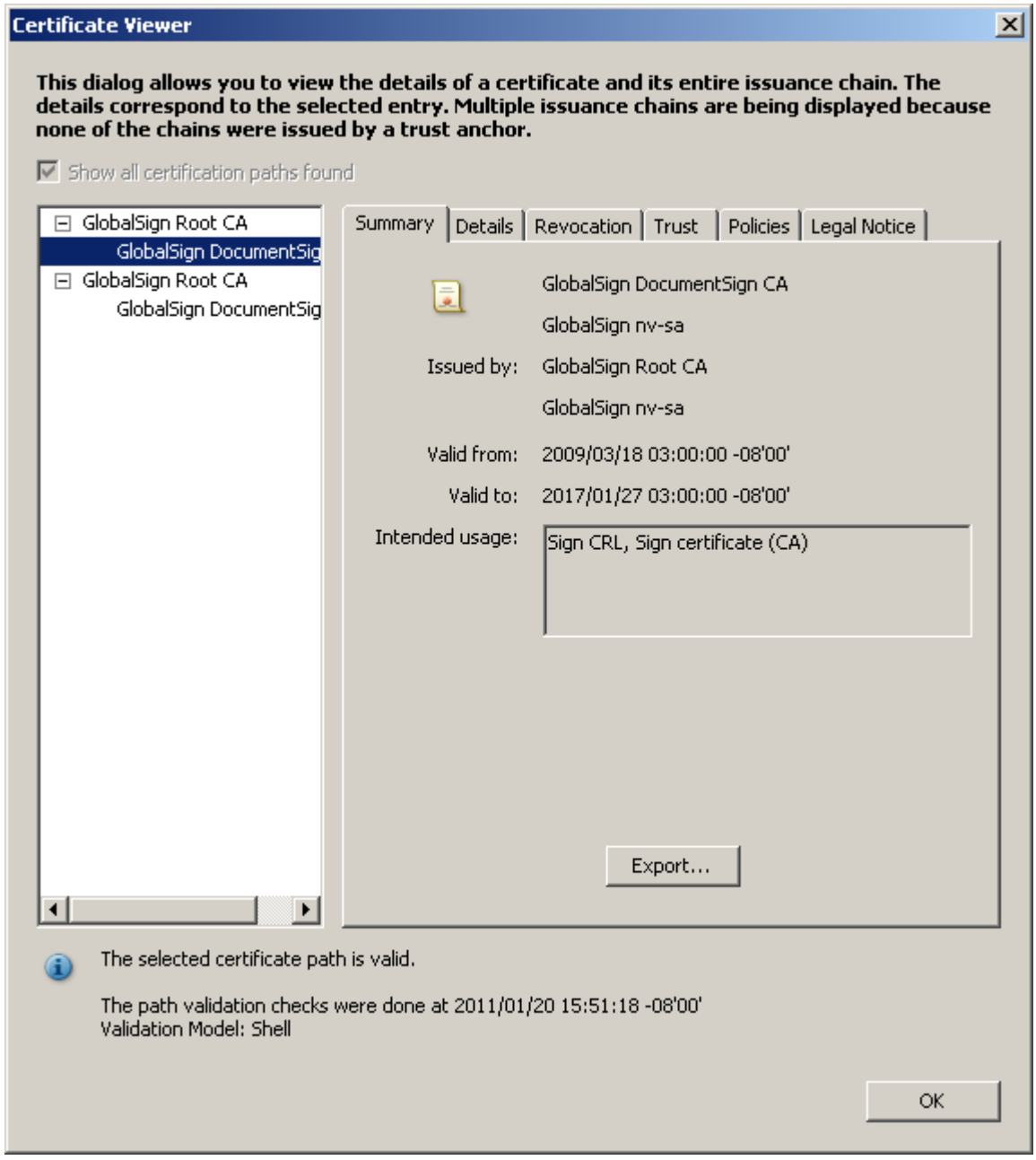
1. Choose **Advanced** (Acrobat) or **Document** (Reader) **> Manage Trusted Identities**.
2. Choose **Certificates** in the **Display** drop-down list.

   Note: In addition to this method, you can also display the certificate from any signature or certificate security method workflow where a **Show Certificate** or **Certificate Details** button appears.

3. Select the certificate.
4. Choose **Show Certificate**. The Certificate Viewer displays the certificate. (Figure 33). The following details are available:
   - **Left hand panel**: The certificate chain.
   - **Bottom area**: A description of the certificate, path validity statement, path validation time, and sometimes the type of validation.
   - **Summary tab**: Owner, issuer, validity period, intended usage. An **Export** button allow users to export the certificate to a file.
   - **Details tab**: Lists all the certificate fields (extensions) and their values.
   - **Revocation tab**: Indicates whether a revocation check occurred and the result. Allows users to initiate a manual check and analyze problems.

- **Trust tab**: Displays the certificates trust level. Provides an **Add to Trusted Identities** button that allows the user to add the certificate to the trusted identity list and set its trust level.
- **Policies tab**: Displays policy restriction information for a signature to be valid, if any.
- **Legal Notice tab**: Displays other certificate policies and a button which links to that policy, if any.

Figure 33 Certificate Viewer



3.5.1.3 Verifying the Identity of Self-Signed Certificates

Certificates are usually issued by a trusted, third-party certificate authority such as VeriSign. However, anyone can set up a certificate authority or create a self-signed certificate purporting to be anyone else. For self-signed certificates or those issued by unknown or untrusted certificate authorities, it is prudent to verify the certificate owner's identity before trusting the certificate.

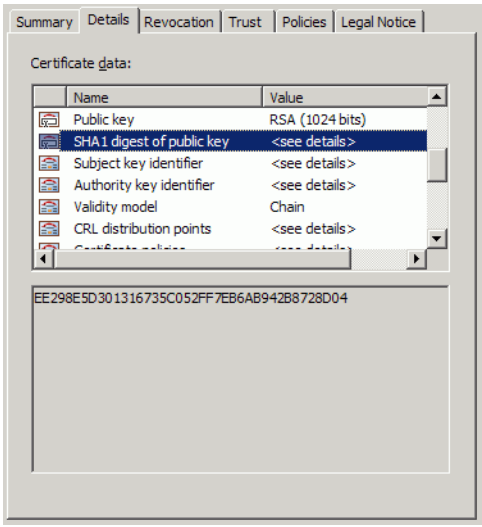
To verify the origin of the certificate:

1. Display the certificate in the Certificate Viewer:
   - If the certificate is embedded in a signature, right click on the signature, choose **Show Signature Properties**, display the Summary tab, and choose **Show Certificate**.
   - If the certificate is already in the Trusted Identify Manager, open it, select Certificates, highlight the requisite certificate, and choose **Show Certificate**.
   - If the certificate is in an FDF file, double-click the attached file, and choose **Certificate Details** in the Import Contact Settings dialog.

     Tip: Double clicking a file other than an FDF will likely install the certificate in the Windows Certificate Store. If the file is .cer, .p7b, or some other format and you want to import into the Acrobat certificate store, save the file locally and import it into the Trusted Identity Manager.

2. Display the Details tab.

Figure 34 Certificates: Verifying originator



3. In the Certificate data panel, scroll to MD5-digest and SHA-1 digest, and note the numbers.
4. Contact the certificate's originator, and verify the MD5-digest and SHA-1 numbers are correct.
5. After the certificate is verified, display the Trust tab and add the certificate to the trusted identities list.
6. Specify certificate trust settings so that it can be used as a trusted root, to certify documents, and so on.

3.5.1.4 Checking Certificate Revocation Status

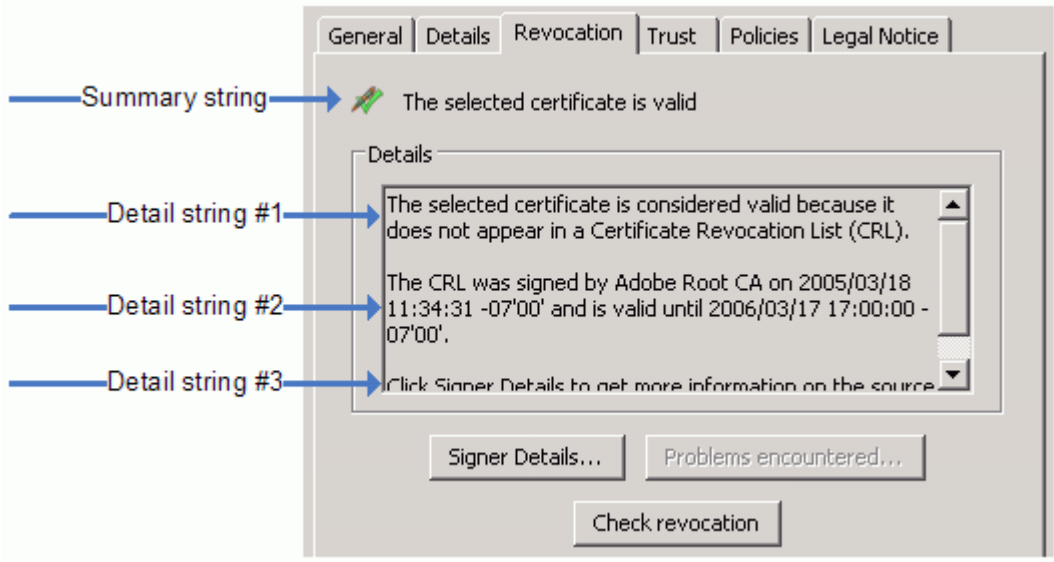
Only the certificate issuer (a certificate authority) has the right to revoke a certificate. A certificate could be revoked because its security might be compromised, it could be invalid for some reason, or the owner of the ID might have left the company. Adobe applications check revocation status as part of its public key authentication.

To check a certificate's revocation status:

1. Choose the Certificate Viewer's Revocation tab.
2. Choose **Check Revocation**.

   The button is only active if revocation checking has not occurred and it is possible to do a check. Revocation checking is not possible for trusted roots and self-signed certificates.
3. Choose **OK**.

Figure 35 Trusted Identities: Viewing revocation status



3.5.1.5 Exporting a Certificate Other than Yours to a File

Users in enterprise settings can send problem certificates to their system administrator or help personnel for troubleshooting. Certificates may be exported from the Trusted Identity Manager. or from the Certificate Viewer.

To do export a certificate from the Trusted Identity Manager:

1. Choose **Advanced** (Acrobat) or **Document** (Reader) **> Manage Trusted Identities**.
2. Choose **Certificates** in the **Display** drop-down list.
3. Select the certificate.
4. Choose **Show Certificate**. The Certificate Viewer displays the certificate. (Figure 33).

5. Choose **Export**.

6. Email the certificate or save it to a file.

3.5.2 Troubleshooting a Document Integrity Problem

If the signature status or overall document status indicates that there could be a document integrity problem, you need to figure out if and how the document changed during the workflow.

. To check document integrity:

1. If the status is unknown and the icon shows a magnifying glass, it is possible signature validation did not occur.
   - Validate the signature(s) as described in "Validating Signatures Manually" on page 40.
   - Verify you have an internet connection and the application is configured properly.
   - Since the problem may not be with your application, try again later.

2. If the status is problematic (displays a warning triangle) shows a, the document has changed but those changes are legal because they have been allowed by the document author. To determine what changed, do any of the following:
   - Open the Signature pane and expand the signature tree to view the change history,
   - Right click on a signature and choose **View Signed Version** or choose **Click to view this version** in the Signature pane to view the version that signed. Review the status for this version. For details, see Chapter 4, "Document Integrity and Preview Mode".
   - Open the **Form Fields Filled In, or Annotations Created or Modified** item to see which fields or annotations were changed or added.
   - Review the document changes as described in "Viewing a List of Post-Signing Modifications" on page 54.
   - Perform a page by page and line by line comparison of the problem version with any of the signed versions. For details, see "Comparing a Signed Version to the Current Version" on page 55.

3. Based on the discovered document changes, determine whether you should trust the document.

4. If the status is invalid (displays a red X), illegal changes have been made to the document, there is no way to undo those changes without further changing the document illegally. Do the following:
   - Contact the sender to resolve the problem. Secure the workflow so that illegal changes are prevented.
   - Policy restrictions on a trust anchor can result in signature invalidity. If you have set a policy restriction, determine if that is the problem remove the restriction.

5. If you still cannot pinpoint the problem, or you need help with some of the steps above, read the following:
   - "Troubleshooting Digital ID Certificates" on page 48

3.5.2.1 LiveCycle Dynamic Forms and the Warning Triangle

Document Integrity Checks for 9.0

Same as 8.1 except that changes to document behavior are detected and invalidate an approval signature: prior versions displayed a yellow triangle upon discovery of changes to document behaviour.

Document Integrity Checks for 8.1

Acrobat 8.1 does not consider all scripts executed during document construction to potentially modify the document, and the detection of a script does not cause Acrobat to flag the document as changed. The application compares the digitally signed and current document versions to determine if the current version has been modified. If there is a change, then a warning triangle appears on the approval signature. For example, the following changes are detected:

- Changes to the value of an LiveCycle form field (including clearing it).
- Changes to the properties of an LiveCycle form field.

Document Integrity Checks for 8.0

Acrobat 8.0 considers all scripts executed during document construction to potentially modify the document even if it's designed with a read-only query or some other "no change" action. The presence and detection of those scripts would trigger Acrobat to display the yellow warning triangle, thereby indicating that the document may have changed. Scripts that invalidate certification would be prevented from executing so it would not be possible for such scripts to invalidate certifying signatures.

3.5.2.2 Viewing and Comparing Changes and Versions

Document authors and recipients often need to know if a document has changed since it was signed. Acrobat keeps track of a document's version number, stores previous document versions in their entirety, and enables users to compare document versions by work and page. When you open a document, the latest version is always displayed whether or not it is the signed version. Document recipients should always remember the following:

- Every time a document is signed, the entire document at the point of signing is stored in the PDF as an incrementally numbered version.
- A signature signs a document at a specific point in time; that is, signature X signs version X and signature Y signs version Y. You can view exactly what the document looked like at that point in time using **View Signed Version**.

For details, see the following:

- Chapter 4, "Document Integrity and Preview Mode".
- "Viewing a List of Post-Signing Modifications" on page 54.
- "Comparing a Signed Version to the Current Version" on page 55.

3.5.2.3 Viewing a List of Post-Signing Modifications

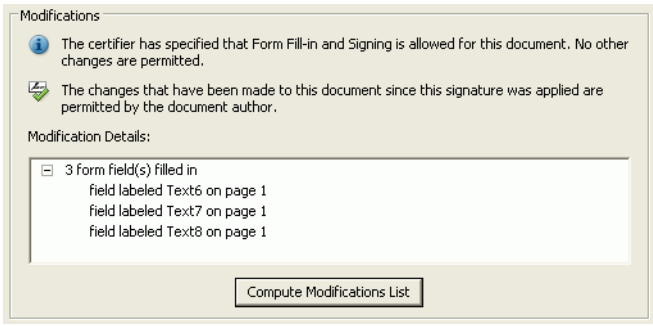
Because it is possible to change a document without changing its appearance, the list of post-signing modifications is often a superset of what is visually displayed when comparing documents using Document Compare. Therefore, a thorough analysis of a signed document's integrity includes viewing the document modifications list.

To view a list of post-signing modifications, open the Signature pane and view the change history. All changes are listed in chronological order.

Note that you can also do the following to get a more condensed list:

1. Right click on a signature and choose **Show Signature Properties**.
2. Choose the Document tab.
3. Choose **Compute Modifications List**. The list is capable of showing the following:
   - Fields (with field names) that have been created, modified, deleted, or filled in.
   - Annotations (comments) that have been created, modified or deleted.
   - Pages that were created, modified or deleted.
   - Spawned pages, deleted spawned pages, and modified spawned pages.
   - Attachments that have been added, deleted, or modified.
   - Miscellaneous: Some changes which occur in memory or cannot be explicitly listed are labelled miscellaneous.

Figure 36 Digital Signature Properties: Modifications panel



3.5.2.4 Comparing a Signed Version to the Current Version

Note: The Compare feature is not available in Adobe Reader.

As you revise a document and save it to a different name or location, you can verify that you have the latest version by comparing it against an older version. If you're revising a document using comments you received during a review, you may need to view a previous version to make sure that you included all the revisions. As a reviewer, you may want to check the updated document against an older version to make sure that the author has incorporated all of your requested changes.

Document Compare does not compare comments or other annotations in the document.

To automatically compare a signed document version with the current view:

1. Right click on a signature and choose **Compare Signed Version to Current Version**.
2. When the two versions appear side by side, review the highlighted areas to review what was changed.

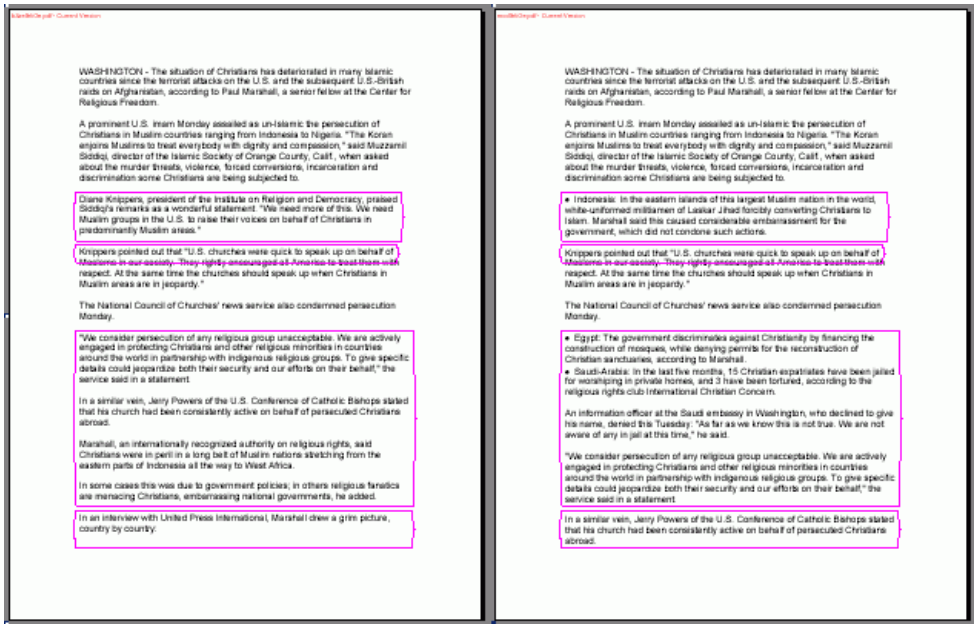
This method compares the two versions page by page. Compare completes by opening a temporary document that summarizes the differences. The first two pages summarize the changed, added, deleted, or moved pages, taking document A as the original and document B as the modified version (Figure 37).

The differences are displayed as follows:

- Even numbered pages (the pages on the left on the two page document view) are pages from document A.
- Odd numbered pages (the pages on the right on the two page document view) are pages from document B.
- Pages that were moved are not shown in the report.
- Any added page, which only exists in document B, is paired with a blank page in the report. Naturally, the added page will be on the right, and the blank page will be on the left in the two-page view.
- Any deleted page, which only exists in document B, is paired with a blank page in the report. Naturally, the deleted page will be on the left, and the blank page will be on the right in the two-page view.
- Pages that were in both documents but were modified are paired with each other in the report. There will be purple hexagons around regions in the two documents that were modified.
- Pages that were not modified will not be in the report.

Figure 37 Compare: By page summary report

Figure 38 Compare: By page



3.6 Document Behavior After Signing

A document's behavior will likely change after it has been signed. Some of it's content may not work (multimedia may not play), some of the application's menu items may be disabled so that you can't use them, and so on. How a document behaves on your desktop could be the result of one or more factors:

- How the document was authored. Were restrictions or requirements placed on the signature fields?
- How a document was signed. Was an approval or certification signature used? Did the signer place restrictions placed on future permissible actions?
- How is your environment configured? Have you trusted the signer's certificate for certain actions? Do you use preview mode?

Note: These items interact in complex ways. In many cases, it's better to accept the application defaults unless instructed to change them by someone knowledgeable about Acrobat's security features.

3.6.1 JavaScript and Dynamic Content Won't Run

High privilege JavaScript and dynamic content in documents will only run if you have explicitly trusted the sender's digital ID certificate for such actions. Because scripts and dynamic content represent a security risk, Acrobat prevents some of those operations by default.

3.6.2 Certifying a Document is Prevented

Only one certification signature is allowed in a document; therefore, it must be the first one.

3.6.3 Form Field Fill in, Signing, and/or Other Actions Don't Work

Once a document is signed or certified, at most only form fill-in, additional signatures, and addition of annotations is allowed. All other operations on the document are disabled as they would invalidate the signature. A certified document may impose further restrictions.

Locking the document after signing also reduces the allowed actions on a document to almost none.

4 Document Integrity and Preview Mode

In general, documents that contain no dynamic content (and only recognizable PDF content) are safer to sign that documents with content that can impair one's ability to see what they are signing and validation. At a high (and simplistic) level, static documents are good; dynamic documents are risky.

Since 8.1, Acrobat has defined PDF features that should be avoided for signable documents that should have a deterministic and repeatable visual rendering. Acrobat's preview mode feature is designed to display that rendering to users during signing and signature validation.

Preview mode analyzes a document for signing best practices and generates a report and messages which indicate the presence of content that might violate those practices. Preview mode suppresses content that may alter the document's appearance, thereby allowing you to view, sign, and validate the document in a static and secure state.

To mitigate the risk associated with signatures in complex documents, preview mode makes the document as static as possible and also generates a report about what behaviors could and could not be suppressed. Using this feature involves the following:

- Learning when and how to use preview mode during signing or signature validation.
- If preview mode or warnings are reviewed, analyzing the result to determine if a document should be trusted.

> **Tip:** There is only a loose correlation between signature or document status and the information displayed in the PDF Signature report. The report is simply a tool designed to identify the presence of potentially malicious dynamic content that could affect the integrity of your signing workflow. Therefore, signature status and PDF signature report information may appear to be contradictory.

4.1 Preview Mode and Signing Workflows

While preview mode may not be needed in your workflow, using it offers a higher degree of assurance that signers and signature validators are viewing the same document and that the signed version is unlikely to change in signing workflows. Users should decide for themselves whether to sign and even trust signatures in documents that contain dynamic content. Therefore, preview mode is recommended for both signing and validation.

Preview mode can be invoked during any part of a workflow:

- **Before or during signing**: When the preference **View documents in preview document mode when signing** is turned on, preview mode is automatically invoked.
- **After signing**: Right click on any signature and choose **View Signed Version**.

Application behavior varies slightly depending on how preview mode was invoked. In general, however, PDF signature report information can be viewed on the document message bar (DMB) text or by choosing the **View Report** button on the DMB.

It may be prudent to not trust a document that generates errors for content that could not be suppressed. For other errors, you may wish to analyze the document for non-conforming content in order to evaluate whether you should trust the document.

4.2 Preview Mode and Validation (View Signed Version)

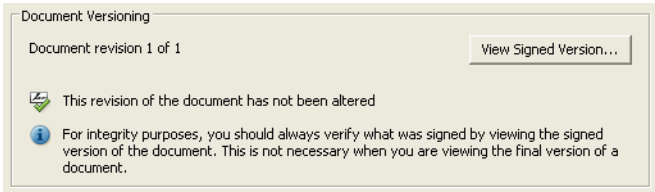
Acrobat and Adobe Reader store in signed documents a unique document version for every signature in the document. In other words, they "remember" that version A is signed, that changes were made to version B, and so on. When you open a document, the latest version is always displayed. However, it is sometimes useful to view the signed version in order to see what content was actually signed.

To view the signed version of a document.

1. Open the Signatures pane, and verify the signature status.
2. Right click on a signature and choose **View Signed Version**.

   View Signed Version is essentially a rollback feature that enables the signature validator to view the document version as ut was at the point in time when it was signed.
3. Choose **View Report** in the Document Message Bar to view a report about the dynamic content that could and could not be suppressed (if any).
4. Right click on the signature and choose Show Signature Properties for more detail about the signature.

Figure 39 Digital Signature Properties: Document Versioning panel



4.3 PDF Signature Reports

Signature workflows often require a document that has a deterministic and repeatable visual rendering that is consistent with the state of the document as signed. This variant does not concern itself with content in a document that is not rendered or does not affect rendering. Examples of such content include metadata and bookmarks.

The level of document's conformance to signing best practices is determined by the presence or absence of certain content. There are three categories:

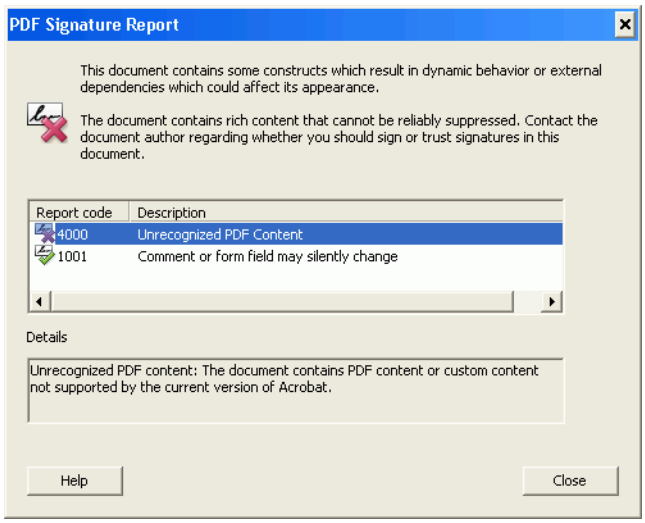
- "Content preview mode cannot suppress" on page 60
- "Content preview mode can suppress" on page 60

- "No external dependencies or dynamic content" on page 61

Content preview mode cannot suppress

Documents that contain content or behaviors which are dynamic or invisible and which cannot be suppressed in preview mode automatically invoke the PDF Signature Report dialog. For example, preview mode cannot suppress (eliminate from the document or make static) externally referenced images, multimedia content outside of the PDF file, and TrueType fonts. Such content is associated with a red X. Users can highlight an error to see more detail (Figure 40). The signer should decide whether or not to trust the document for signing. If you are not the author, contact the author for additional information.

Figure 40 PDF Signature Report: Content which cannot be suppressed in preview mode



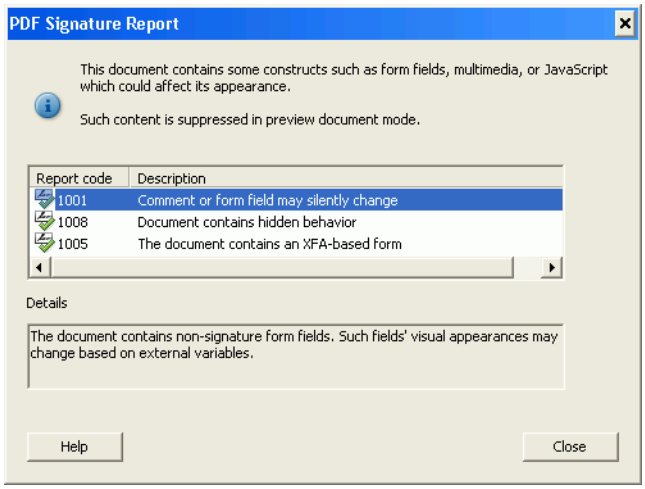
Content preview mode can suppress

Preview mode can suppress certain types of dynamic content. When documents are signed in preview mode, the signed view of the document (with the dynamic content suppressed) is saved so that signature validators can use the View Signed Version feature to validate the signature and see what the signer saw when they were signing.

The **View Report** button opens a dialog that lists discovered rich content. Suppressed content is associated with a green check (Figure 42). If you are concerned about the presence of rich content even though it is suppressed in preview mode, review the error codes and descriptions in the PDF Signature Report dialog for more information.

Figure 41 Document Message Bar: Suppressible rich content

Figure 42 PDF Signature Report: Suppressed content



No external dependencies or dynamic content

For the highest level of document integrity insurance, do not allow dynamic content or any content with external dependencies. These documents are the safest to sign because they can be reliably displayed and do not require a special viewer or preview mode to be signed or to later view what was signed.

Figure 43 Document Message Bar: No dynamic content message

4.4 Signature Report Error Codes

As shown in the following tables, report errors categorize content as one of the following:

- **Dynamic features**: Presentations, user-launched multimedia, JavaScript, dynamic forms, and so on.
- **PDF content with variable rendering**: JavaScript, non-embedded fonts, and so on.
- **External content**: Hyperlinks, alternate images, linked files, and so on.
- **Uncategorized content**: Unrecognized or malformed PDF content.

Table 9 dynamic feature warnings

String	Code	Description
Document contains hidden behavior	1000	The document contains hidden actions that may not be intended or known by the end user. Actions include JavaScript actions (document open, save, etc.), playing multimedia, executing a menu item, and so on.
Comment or form field may silently change	1001	The document contains non-signature form fields. Such fields' visual appearances may change based on external variables.

Table 9 dynamic feature warnings

String	Code	Description
Comment or form field may silently change	1002	The document contains comments. Comments' visual appearances may change based on external variables.
Document may silently launch menu items	1003	The document contains named actions that may launch menu items without the user's knowledge.
Presentation elements may change appearance	1004	Presentations are not allowed since a presentation may contain animations or other elements that may change document appearance or behavior.
The document contains a dynamic form	1005	XFA-based (dynamic forms) documents are not allowed since such forms could alter the document's appearance or behavior.
Document contains links to external PDFs	1006	The document links to external PDFs on the Internet, file system, or network and it has no control over the nature of that linked content. Embedded Go-To actions must not refer to external hierarchies.
Comment or form field may silently change	1007	Disallowed annot type: <annot type>. One or more form fields are associated with a 3D object, file attachment, multimedia, or other dynamic objects.
Document contains hidden behavior	1008	Disallowed action type: <action type>. The document contains hidden actions that may not be intended or known by the end user. Actions include JavaScript actions (document open, save, etc.), playing multimedia, executing a menu item, and so on.
Document contains hidden behavior	1009	The document's content is divided into layers that can be silently displayed or hidden on the fly.

Table 10 PDF Content with variable rendering

String	Code	Description
Page content may silently change	2004	Visual elements may change based on external variables. For example, a logo may change color based on time or zoom level. No postscript XObjects allowed.
Document may not open in the future	2006	Some or all of the content is encrypted and the encryption method is not available in standard Acrobat installations. For example, the document may be protected by the Adobe Policy Server. Document contain streams encrypted using crypt filter.
Page content may silently change	2007	The document author has enabled image interpolation. No image interpolation is allowed.
Page content may silently change	2009	The document uses a PDF transfer function that interprets and replaces color. For example, it could replace black with white. Extended graphic state should not use the TR key
Page content may silently change	2010	The document uses a PDF transfer function that interprets and replaces color. For example, it could replace black with white. If present, the extended graphic state's TR2 key must be set to default
Page content may silently change	2011	The document's extended graphic state uses the FL key. The key is a number that indicates how much flatness tolerance should exist when drawing objects. Content may display differently from Acrobat to other applications
Page content may silently change	2012	Image XObject must not contain an alternate version

Table 10 PDF Content with variable rendering

String	Code	Description
Text appearance may silently change	2013	Document contains non-embedded fonts. When the document opens on a system that does not have the requisite fonts, Acrobat will replace them with some other font. Users should always turn on font-related warnings. The non-embedded fonts warning is turned off by default. It can be turned on by setting the DigSig\bEnNonEmbFontLegPDFWarn preference to true. The disallowed font type warning is also turned off by default and can be turned on by setting the DigSig\bTrueTypeFontPDFSigQWarn preference to true. These are Windows registry or Mac plist settings. See the *Security Administration Guide* for more details.
Text appearance may silently change	2014	Disallowed font type: <font type>. True type and TrueType-based OpenType fonts are not allowed because they are programs and may change the document's appearance based on external variables.

Table 11 External Content

String	Code	Description
Document links to external content	3000	Document links to images not in the PDF. No external XObjects allowed.
Document links to external content	3001	Document links to images not in the PDF that are used as alternates. For example, an alternate, high resolution images might be specified for printing. Images must not contain an OPI alternate version.
Document links to external content	3002	Document contains external streams. The author has flagged some PDF bytes as a stream which may get data from an external source.
Document links to external content	3003	Document links to images not in the PDF that are used as alternates. For example, an alternate, high resolution images might be specified for printing. Form XObject must not contain an OPI alternate version.

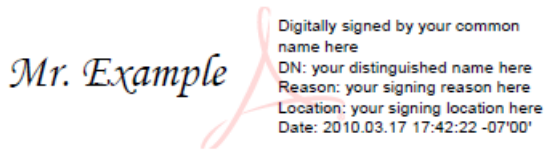
Table 12 Uncategorized warnings

String	Code	Description
Unrecognized PDF content	4000	Unrecognized PDF content: The document contains PDF content or custom content not supported by the current version of Acrobat. The document may have been created by a later version of Acrobat.
Page content may silently change	4001	Unrecognized drawing operator: The document contains PDF content or custom content not supported by the current version of Acrobat. The document may have been created by a later version of Acrobat.
PDF content contains errors	4002	Malformed drawing instructions: Syntax error. Page content violates the grammar for page content definition. For example, the instruction might specify drawing a square but the syntax for doing it is incorrect.

5 Customizing Signature Appearances

Personalized signature appearances allow you to provide additional information about yourself, affiliations, or company. Home users often modify the default appearance as a matter of personal taste, while enterprise users are sometimes provided with a company-specific style and logo.

Every aspect of the visible appearance is subject to customization, and the options range from using logos, a replica of your handwritten signature, photos, images, and text. A typical appearance consists of three components, and each can be separately customized:



- **Signature**: A graphic that identifies the signer on the left-hand side of the appearance, such as a photo or scanned signatures. Transparent backgrounds allow a watermark to be visible in the underlying layer.
- **Signature details**: Signature data that the signer wants to appear to the right of the signature.
- **Watermark or logo**: An image that appears behind the signature. The default is the PDF logo.

You can create any number of appearances ahead of time for later use, and the signing workflow allows you to select one from your library appearances. However, you can also just use the default appearance created from your name or create one on-the-fly at signing time.

When you sign a PDF file, the appearance becomes part of the signed document. It is not part of the signature.

PDF/A compliance

PDF/A is a subset of PDF intended as a file format suitable for the long-term archiving of electronic documents. Appearances in PDF/A-compliant documents must be must conform to the following:

- Embed fonts. PDF/A requires font embedding because font metrics and glyph compliments change over time. Long term archiving can not rely on anything not included in the document.
- Colors should use DeviceRGB when signing RGB/Office docs or else with associated ICC profiles.

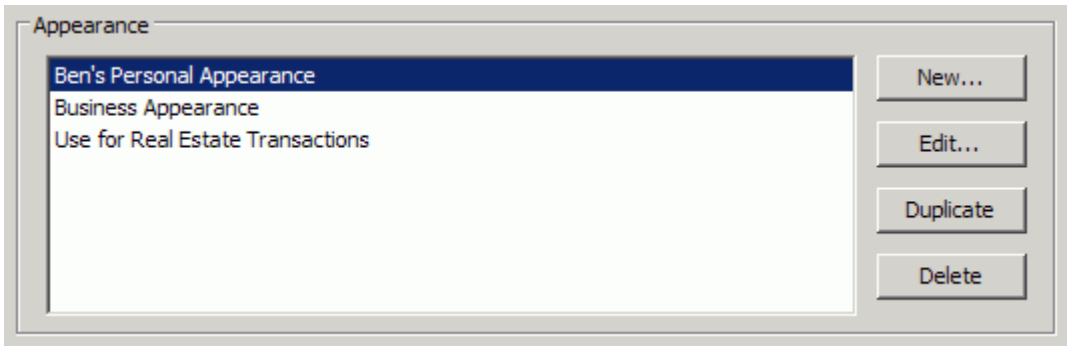
5.1 Customizing a Signature Appearance

Users generally customize one or more signature appearances and store them for later use. Available signatures are listed in the appearance panel at signing time (Figure 44).

To customize a new signature appearance:

1. Choose **Edit > Preferences** (Windows) or **Acrobat > Preferences** (Macintosh).
2. Choose **Security** in the left-hand list.
3. In the Appearance panel, choose **New** or **Edit**.

Figure 44 Signature appearance library

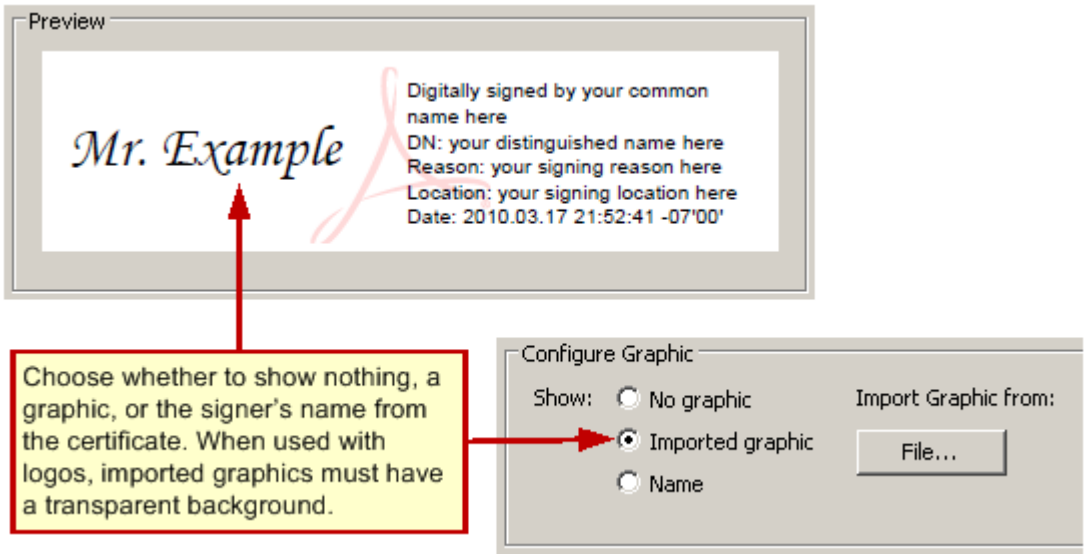


4. Configure the signature appearance **Title**: Any title used to identify the appearance. These are useful for identifying which appearance to use for particular workflows.

Figure 45 Signature appearance: Title

5. Set the graphic options in the Configure Graphic panel
   - **No graphic**: No graphic is used.
   - **Imported graphic**: Choose **File > Browse**, select a file and choose **OK**.
   - **Name**: Your text extracted from the signing certificate.

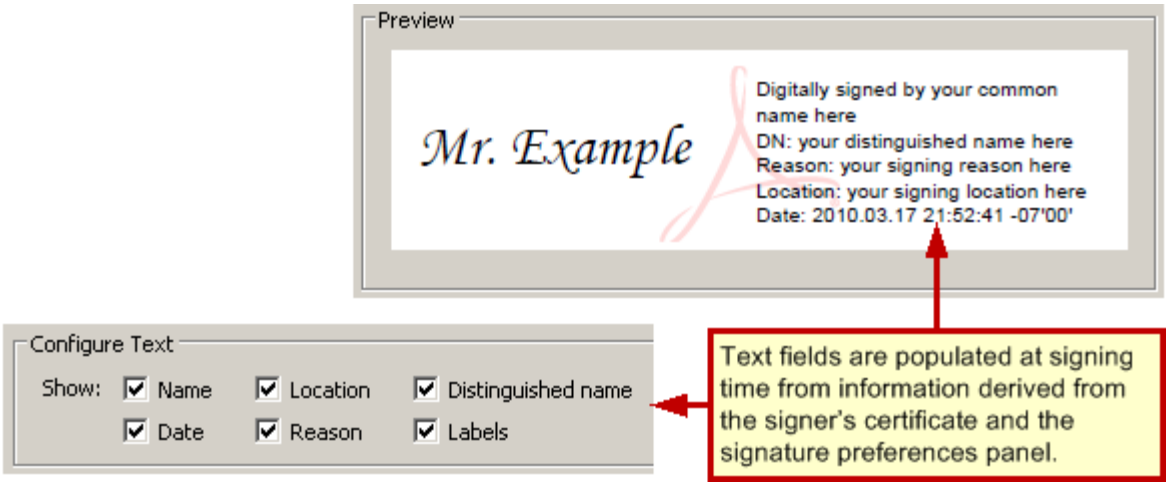
Figure 46 Signature appearance: Import graphic



6. Set the text fields to display in the appearance in the Configure Text panel:
   - **Name**: The name associated with the certificate.

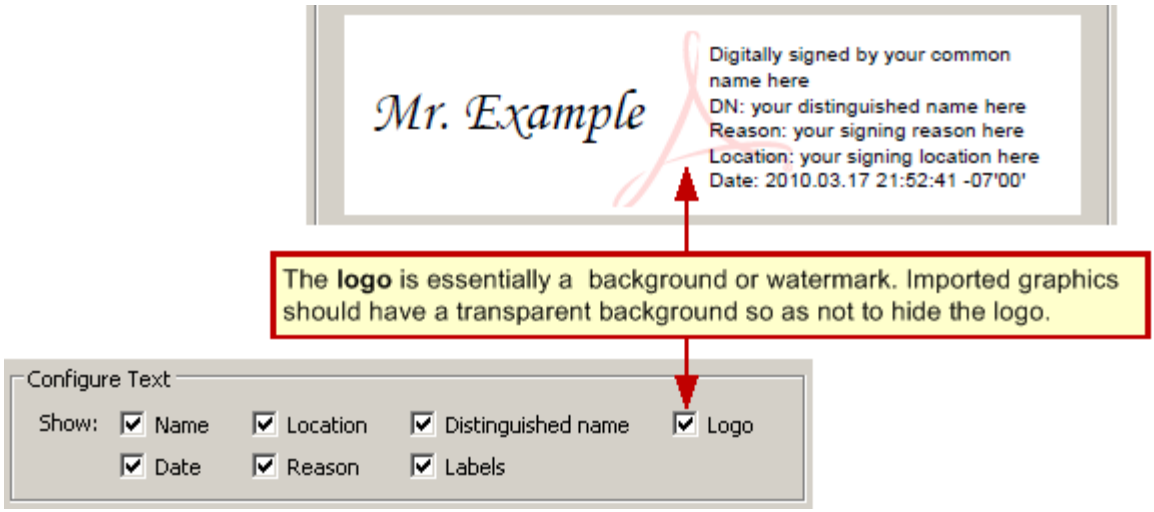
- **Date**: The date signed. Signature appearances can only display local (computer) time, and it will likely differ from that in the Date/Time tab on the Signature Properties dialog when a timestamp server is used.
- **Location**: The location associated with the identity configured in Acrobat.
- **Reason**: The reason for signing.
- **Distinguished name**: A name with details such as country, organization, organizational unit, and so on.
- **Labels**: A label for each of the items above. For example, `Reason`.

Figure 47 Signature appearance: Text fields



7. **Logo**: The logo or graphic used as a background watermark. The default watermark is the Adobe PDF logo. Your watermark will be shared automatically in all of your signature appearances.

Figure 48 Signature appearance: Logo or watermark

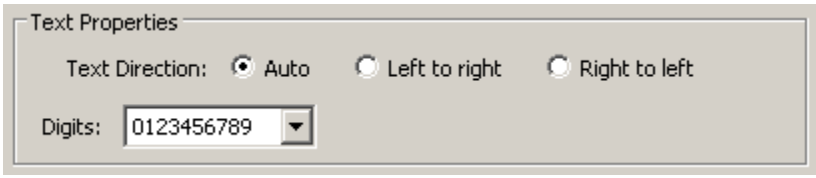


8. Set the text direction and character set in the Text Properties panel:
   - **Text Direction**: Choose a direction appropriate for the signer's language.

- **Digits**: If languages are installed that use digits other than 1234567890, the drop-down list will be populated with alternate choices. Choose a digit set appropriate for the signer's language.

9. Choose **OK**.

Figure 49 Signature appearance: Text properties



5.2 Changing the Logo (Watermark or Background)

A watermark is a partially transparent graphic or logo that appears "behind" a signature. By default, the watermark is the Adobe PDF logo. Line (vector) art that is simple and unobtrusive often works best.

1. Import a logo or create a new one in a program such as Adobe Illustrator.
2. Illustrator instructions: Set a low transparency level and flatten the transparency:
    1. Select all and group the objects if there is more than one.
    2. Choose **Window > Transparency** and move the slider to some low value such as 20%.
    3. Choose **Object > Flatten Transparency**. Leaving the Raster/Vector balance at 100%.
    4. Save the file as a PDF.
3. Open the PDF file in Acrobat.
4. Crop the page and remove white space.

    Note: The method varies across product versions. For example, for 8.x, choose **Document > Crop Page** and check **Remove White Margins**.

5. Save the file as SignatureLogo.pdf in:
    - **Windows**: C:\Documents and Settings\<user>\Application Data\Adobe\Acrobat\<version>\Security.
    - **Macintosh**: \Users\<user name>\Library\Application Support\Adobe\Acrobat\<version>\Security.

5.3 Creating Appearances from Handwritten Signatures

You can use your handwritten signature as the graphic displayed by your signature appearance as follows:

1. Scan your signature at a relatively high DPI; for example, 300.
2. Convert the image to vector drawing.
3. Make the background transparent.
4. Save the image in the desired format. A wide variety of formats is supported.

Figure 50 Handwritten signature appearance

6 Controlling Signing with Seed Values

Acrobat's seed value feature helps authors control document behavior once it has been routed to the signer. Seed values can be used to embed certificate requirements and other instructions in signature fields. When a signers signs a custom, "seeded" field, the author-specified behaviors are automatically invoked and enforced.

A seed value specifies an attribute and attribute value. The author can make the seed value a preference or a requirement. For example, You can use seed values to limit a user's choices when signing a particular signature field. For details about what seed values can be used to control, see the following:

6.1 Seed Value Basics

How you add a seed value to a document varies across products:

- **Acrobat**: JavaScript calls must be used because no direct user interface is provided. The JavaScript function `signatureSetSeedValue` sets properties that are used when signing signature fields. The properties are stored in the signature field and are not altered when the field is signed, the signature is cleared, or when `resetForm` is called. This method (and JavaScript generally) can be executed with a batch process, by dropping the script in Acrobat's JavaScript subdirectory, menu events, Acrobat's JavaScript debugger, and other methods.
- **LiveCycle Designer**: Set seed values via the signature field Properties panel.

When setting seed values, keep in mind the following:

- Seed values should not be set on signed documents and cannot be set on certified documents after the document is certified. They are primarily used to configure fields on documents that are not yet signed.
- Setting a seed value often causes Acrobat to not display or use its default settings. For example, default reasons are stored in a registry list, and specifying signing reasons with a seed value overrides that list.
- Seed value properties include those listed in Table 5. Note that `certspec` and `timeStampspec` are objects that have multiple properties.

6.1.1 Changes Across Releases

Each Acrobat release results in support for additional seed values as shown in Table 2.

6.1.2 Supported Seed Values

Note: The examples in this document demonstrate the simplest case. For more information, refer to the *Acrobat JavaScript Scripting Guide* and *JavaScript for Acrobat API Reference*.

Table 5 Seed values: Object properties and descriptions

Property	Type	Description
certspec	object	A seed value CertificateSpecifier Object. For details, see "Specifying Certificate Properties for Signing" on page 83.
digestMethod	array of strings	(Acrobat 8.0) An array of acceptable digest methods to use while signing. These are only applicable if the digital ID contains RSA public and private keys. If they contain DSA public/private keys, then the value is always SHA1. Valid values include: MD5, SHA1 (default), SHA256, SHA384, SHA512, and RIPEMD160.
filter	string	The language-independent name of the signature handler to be used when signing.
flags	number	A set of bit flags controlling which of the following properties are required. The value is the logical OR of the following values, which are set if the corresponding property is required: **1**: filter **2**: subFilter **4**: version **8**: reasons **16**: legalAttestations (Acrobat 8.0) **32**: shouldAddRevInfo (Acrobat 8.0) **64**: digestMethod (Acrobat 8.0) **128**: lockDocument **256**: appearanceFilter Usage: 1 specifies filter, 3 specifies filter and sub-filter, and 11 specifies filter, sub-filter, and reasons. If this field is not present, all properties are optional. For more details, refer to the *PDF Reference*.

Table 5 Seed values: Object properties and descriptions

Property	Type	Description
version	real	See version (above). (Optional) The minimum required capability of the signature field seed value dictionary parser. A value of 1 specifies that the parser must be able to recognize all seed value dictionary entries specified in PDF 1.5. A value of 2 specifies that it must be able to recognize all seed value dictionary entries specified in PDF 1.7 and earlier. A value of 3 specifies that it must be able to recognize all seed value dictionary entries specified in PDF1.7-ADBE-3 and earlier. The Ff (flags above) entry indicates whether this is a required constraint. For more details, refer to the *PDF Reference*.
legalAttestations	array of strings	(Acrobat 7.0) A list of legal attestations that the user can use when creating an MDP (certified) signature.
mdp	string	(Acrobat 7.0) The modification, detection, and prevention (MDP) setting to use when signing the field. Values include: allowNone default defaultAndComments While `allowAll` is a legal value, it cancels out the effect of `mdp` and no certification signature can be used for this field.
reasons	array of strings	A list of reasons that the user is allowed to use when signing. (Acrobat 8.0) If this array contains a single empty string and reasons are marked as required using the flags variable, Acrobat will not allow a signing reason. If this array is empty and reasons are marked as required, an exception will be thrown.
shouldAddRevInfo	boolean	(Acrobat 8.0) The default value is false. If true, the application does certificate and chain revocation checking and embeds the information in the signature. If true and the flag is set to require these actions, any failure in these actions results in signing failure. Only relevant if subFilter is adbe.pkcs7.detached or adbe.pkcs7.sha1. If the subFilter is adbe.x509.rsa_sha1 and adding revocation information is required, the signing operation fails.
subFilter	array of strings	An array of acceptable formats to use for the signature. Refer to the Signature Info object's `subFilter` property for a list of known formats.
timeStampspec	object	(Acrobat 7.0) A seed value timeStamp specifier object. It uses the `url` and `flags` properties to specify a timestamp server. For details, see "Specifying Timestamps for Signing" on page 79
version	number	The minimum required version number of the signature handler to be used to sign the signature field. Valid values are 1 and 2. (Acrobat 8) This must be set to 2 if this seed value object contains any Acrobat 8-specific content that is marked as required.

Table 5 Seed values: Object properties and descriptions

Property	Type	Description
lockDocument	name	(Optional; PDF 1.7-ADBE-3) Allows the author to add a Lock Document checkbox to the signing dialog so a signer can lock the document at the time of signing. The default is `auto`. • **true**: Indicates that the desired action is that the document should be locked at the time of signing. If the Ff entry indicates that LockDocument is not a required constraint and that the user can choose to override this at the time of signing. Otherwise the document must be locked after signing. • **false**: A false value indicates that the document should not be locked after signing. Again, the required flag determines whether this is a required constraint. • **auto**: The auto value allows the consuming application to decide whether or not to present the lock UI for the document and whether to honor the required flag based on the properties of the document.
AppearanceFilter	string	(Optional; PDF 1.7-ADBE-3) A text string naming the appearance to be used when signing the signature field. Conforming readers may choose to maintain a list of named signature appearances and this text string provides authors with a means of specifying which appearance should be used to sign the signature field. If the required bit in Ff is set (see flag above), then the appearance must be available to sign the document and must be used.

6.1.3 Enabling JavaScript to Set Seed Values

Authors sometimes use JavaScript to set seed values for signature fields. When Acrobat's JavaScript console is used for JavaScript execution, the JavaScript debugger must be enabled.

Tip: If you do not intend to set seed values with JavaScript through Acrobat's JavaScript debugger, skip this section.

To enable the JavaScript debugger:

1. Choose **Edit > Preferences** (Windows) or **Acrobat > Preferences** (Macintosh).
2. Choose **JavaScript** in the left-hand category list.
3. Check **Enable JavaScript**.
4. Check **Enable JavaScript debugger after Acrobat is restarted**.
5. Restart Acrobat.

To set seed values with the console (JavaScript debugger) in Acrobat, do the following:

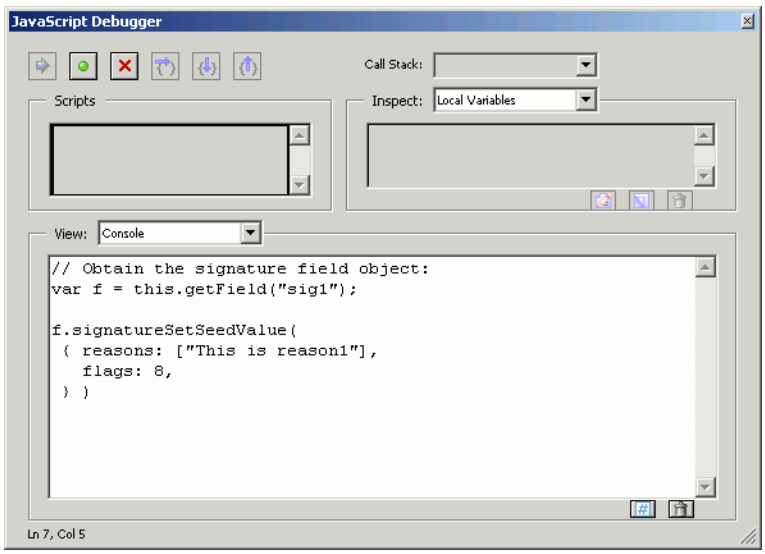
1. Choose **Ctrl + J**.
2. Use the **View** drop-down list and select Console.
3. Enter the requisite JavaScript.
4. Highlight the JavaScript. If you do not highlight the JavaScript, only the last line of code is executed.

5. Press **Control + Enter** simultaneously or select the **Enter** key on the numeric keypad.

   Tip: When the JavaScript is executed correctly, the debugger returns "undefined."

6. Save the document, and test the field.

Figure 51 Seed values: JavaScript debugger



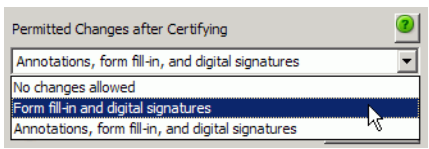
6.2 Forcing a Certification Signature

By default, signature fields can be signed with an approval or certification signature at the time of signing. However it is possible to constrain a signature field such that only a certification signature can used.

Certification signatures are always associated with modification detection and prevention settings (and an `mdp` property) that control what types of changes can be made to a document before the signature becomes invalid. Changes are stored in the document as incremental saves beyond the original version of the document that was covered by the certifying signature. The `mdp` seed value allows you to control what behavior the signer can allow after signing (Figure 52).

Note: If a document is already signed, fields with the `mdp` property specified will NOT invoke the certifying workflow. No error is given. Do not use `mdp` unless you are sure the requisite field will be the first one signed.

Figure 52 Seed value: Forcing mdp selection during certification



MDP has one of the following four values:

- **allowAll**: *Do not use* `allowAll` unless you want to force an *approval* signature since this value results in MDP not being used for the signature and therefore doesn't force a certifying signature.
- **allowNone**: Document changes invalidate the signature and lock the author's signature. `allowNone` bypasses any custom `legalAttestations` because no document changes can occur and the user does not therefore need to be warned about malicious content. Do not use with `legalAttestations`.
- **default**: Allow form field fill-in if fields are present in the document as well as additional signatures. Other changes to the document invalidates the signature.
- **defaultAndComments**: Allow form field fill-in if fields are present in the document and allows annotations (comments) to be added, deleted, or modified as well as additional signatures. Other changes to the document invalidates the signature. Note that annotations can be used to obscure portions of a document and thereby affect the visual presentation of the document.

To force a certifying signature for a particular field:

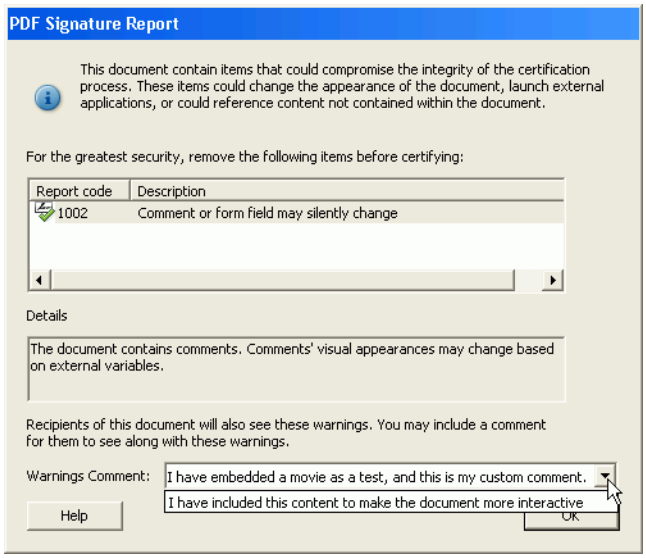
1. Create a signature field with an intuitive name and tooltip.
2. Create the JavaScript that gets the field object and uses the seed value method (Example 6.1).
3. Set the `mdp` value:
   - **allowNone**: Do not use with `legalAttestations`.
   - **default**: Allow form field fill-in, including signing.
   - **defaultAndComments**: Allow form field fill-in, signing, and comments.
4. Add `legalAttestations` if you would like to customize user choices.

Signers can view warnings about potentially malicious content (content that could change the appearance of a signed document) during signing. The Review button in the signing dialog runs the PDF/SigQ Conformance Checker which reports on rich content. Signers can then enter a **Warnings Comment** in the drop-down list indicating why that content is OK.

When specifying custom legal attestations, keep the following in mind:

- Since certified document warnings only appear in certifying workflows, only use `legalAttestations` if you also use `mdp`. For details, see "Forcing a Certification Signature" on page 74.
- Customizing legal attestations overrides and removes default choices for the signer.
- Custom text is viewable in the user interface during signing when the signer chooses **Review** in the signing dialog.

Figure 53 Seed values: Custom legal attestations



5. Highlight the JavaScript and choose **Control + Enter** (or the **Enter** key on the numeric keypad).

   When someone signs the field, the certifying workflow is invoked and only the specified **`mdp`** settings will be available (Figure 52).

6. Run the JavaScript, save the document, and test the field.

Example 6.1: Seed value: mdp

```
// Obtain the signature field object:
var f = this.getField("mySigFieldName");

f.signatureSetSeedValue(
 {
    mdp: "defaultAndComments",
    legalAttestations: ["Approved by Management", "Signed by Procurement"]
 } )
```

6.3 Giving Signers the Option to Lock a Document

While certifying a document allows authors and signers to restrict certain document features, for example, subsequent signing and filling out forms, these permissions are set at the document level and cannot become more restrictive as signatures are applied. Acrobat 9 provides a seed value that adds a **Lock Document** checkbox to the signing dialog when the configured field is selected for signing. This allows any signer to completely lock the document from further changes.

The author is given the choice of requiring their seed value to be honored by the signing application or to be an optional recommendation. This is provided through the use of the required flags bitfield to indicate whether or not the seed value is optional or mandatory. The options are as follows:

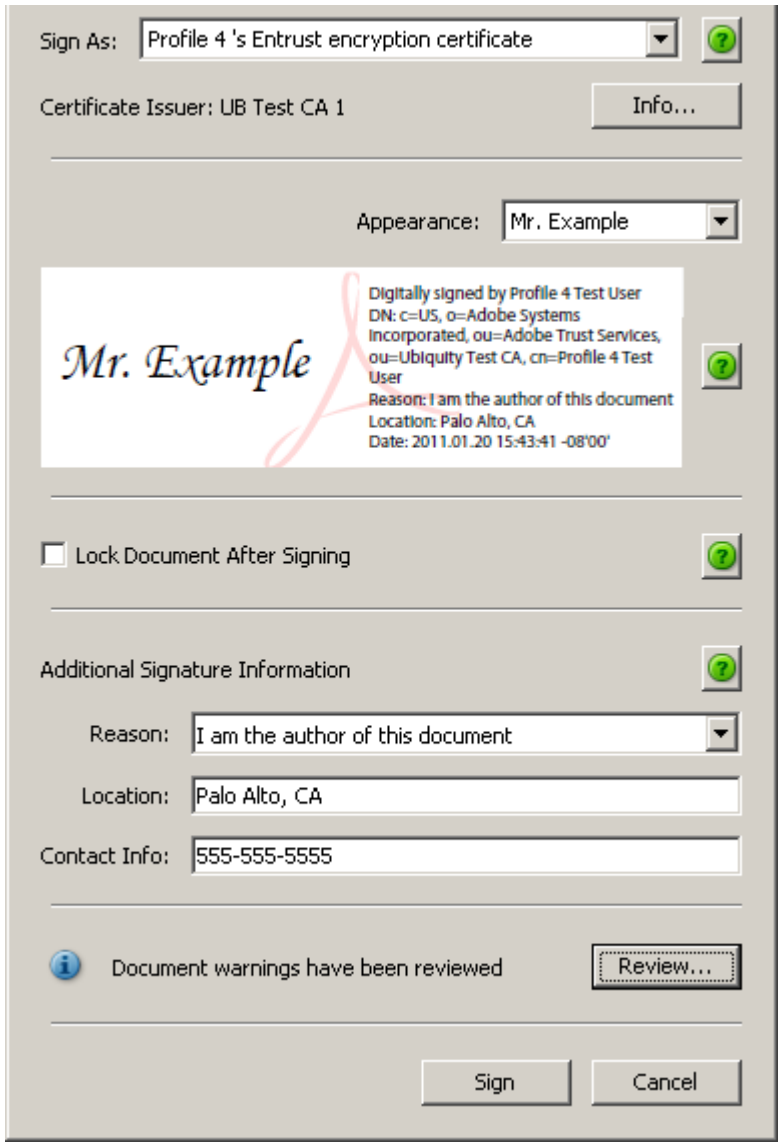
- **true**: Indicates that the desired action is that the document should be locked at the time of signing. If the Ff entry indicates that LockDocument is not a required constraint, the user can choose to override this at the time of signing. Otherwise the document must be locked after signing.
- **false**: A false value indicates that the document should not be locked after signing. By default, the user is given the option of locking after signing. Again, the required flag determines whether this is a required constraint.
- **auto**: Default. Allows the consuming application to decide whether or not to present the lock UI for the document and whether to honor the required flag based on the document properties.

Example 6.2: Seed value: lockDocument

```
f = this.getField("mySigFieldName");
f.signatureSetSeedValue({lockDocument:<true/false/auto>});

//Set the setting as required
f.signatureSetSeedValue({lockDocument:<true/false/auto>, flags:0x80});
```

Figure 54 Sign Document dialog: With Lock Document checkbox added



6.4 Forcing Signers to Use a Specific Signature Appearance

Enterprises and other structured work environments sometimes provide uses with predefined signature appearances. These appearances are then used for specific signing purposes. For example, the appearance may identify the signer's organizational affiliation or a particular task or workflow that pertains to the signed document.

Authors in such environments can specify which signature appearance is required for any given signature field. As with other seed values, a flag bit is used to indicate whether or not the field is a recommendation or mandatory. A signature field is correlated with a specified name which is then used to match a given appearance. The string name must exactly match the name of a signature appearance for it to be selected.

Example 6.3: Seed value: signatureAppearance

```
f = this.getField(<Field Name>);
f.signatureSetSeedValue({AppearanceFilter:"Example Appearance Name"});

//Set the setting as required
f.signatureSetSeedValue({AppearanceFilter:"Example Appearance Name", flags:
0x100});
```

6.5 Adding Custom Signing Reasons

Acrobat predefines several common signing reasons such as "I am approving this document." However, the author can specify custom reasons and make those reasons required or optional. When custom reasons are marked as required, users cannot enter any new reasons as the field becomes read-only. When those reasons are flagged as optional, signers can choose one of the provided reasons or create a new one by typing in the **Reason** field. Specifying a signing reason will remove all of the default reasons from the reason drop-down list.

User interface impact: Note that end users have a user interface preferences that allows them control whether or not the reason's field appears. The preference interacts with the reasons flag as shown in Table 6, and the logic is as follows:

- The document author has control over whether the UI appears and the `required` flag overrides user-specified settings.
- When a flag makes the field `optional`, end users can enter custom reasons.

To specify custom signing reasons:

1. Create a signature field with an intuitive name and tooltip.
2. Create the JavaScript that gets the field object and uses the seed value method (Example 6.4).
3. Add the reasons. The reason list is an array in the format of ["one", "two", "three"].
4. Enter a flag value to indicate whether the value is required or not.
    - If a reason is not required, signers can add their own custom reason while signing.

- If the predefined reasons are required, signers are prevented from saving a document with their own reason (Figure 55).

5. Run the JavaScript, save the document, and test the field.

Table 6 Reason field behavior

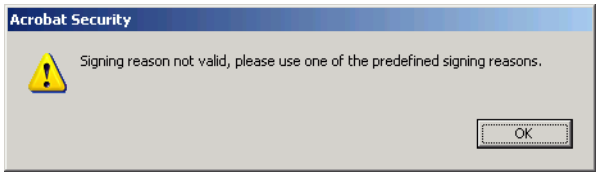
# of Reasons	UI Pref	Flag	Reason Behavior
0 (empty array)	off	Required	Reason field does not appear in UI.
0 (empty array)	on	Required	Reason field does not appear in UI.
0 (empty array)	off	Optional	Reason field does not appear in UI.
0 (empty array)	on	Optional	Display the default list.
1 or more	off or on	Required	Display the custom reasons in a read-only field.
1	off	Optional	Reason field does not appear in UI.
2 or more	on	Optional	Display the custom drop-down list and let the user enter a custom reason.
2 or more	off	Optional	Reason field does not appear in UI.

Example 6.4: Seed value: Custom signing reason

```
// Obtain the signature field object:
var f = this.getField("mySigFieldName");

f.signatureSetSeedValue(
 { reasons: ["This is a reason", "This is a better reason"],
   flags: 8
 } )
```

Figure 55 Seed value: Reason not allowed error



6.6 Specifying Timestamps for Signing

Timestamps originating from a timestamp authority's timestamp server are often associated with signatures. If it is critical in your workflow to acquire a secure timestamp with a digital signature, it can be controlled at the document level instead of relying on the signer's Acrobat configuration. Adding a seed value to the signature field with the timestamp server authority settings overrides the corresponding application level settings, if any. Use the **`timeStampspec`** specifier object's `url` and `flags` properties to specify a timestamp server.

Table 7 Seed values: timeStampspec properties

Property	Type	Description
url	string	URL of the timeStamp server providing a RFC 3161-compliant timeStamp.
flags	number	A flag controlling whether the time stamp is required (1) or not required (0). The default is 0.

To specify a timestamp server:

1. Create a signature field with an intuitive name and tooltip.
2. Create the JavaScript that gets the field object and uses the seed value method (Example 6.5).
3. Provide a URL for the `timeStampspec` object.

   Tip: Timestamp seed value settings override the end users' application settings, if any.

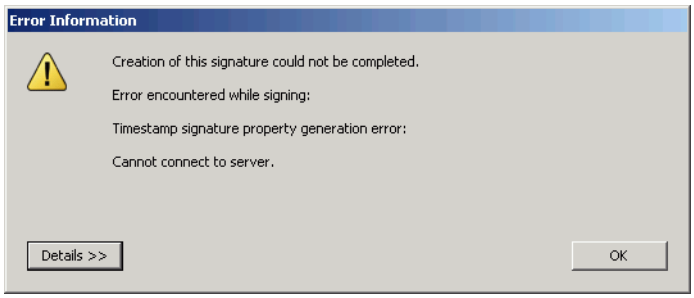
4. Enter a flag value to indicate whether the value is required or not.
   - If it is required, the field is automatically timestamped on signing. If the application cannot find the server, an error appears (Figure 56).
   - If it is not required, the field will is automatically timestamped on signing if the application can find the server. If it cannot find the server, the signature is not timestamped and no error appears.
5. Run the JavaScript, save the document, and test the field.

Example 6.5: Timestamp server seed value

```
// Obtain the signature field object:
var f = this.getField("mySigFieldName");

f.signatureSetSeedValue(
 {
   timeStampspec: {
     url: "http://153.32.69.130/tsa",
     flags: 1
   }
 } )
```

Figure 56 Time stamp server error



6.7 Specifying Alternate Signature Handlers and Formats

Organizations may choose to use alternate signature technologies or implementations (signature handlers), provided by third party software developers. For example, a corporation may have deployed Entrust Entelligence® to all their desktops and may choose to use the Entrust signature plug-in with Acrobat. Two seed values allow authors to specify which signature handler and format to use. By using a standard format, interoperability across multiple signature handlers is possible.

Filter also allows authors to control what handler version is required. For example, for Acrobat 6.x, the PPKLite version is 0. For Acrobat 7.x, the PPKLite version is 1. Therefore, specifying a version of 1 prevents signers from signing when their application is older than Acrobat 7.0. Custom handlers can use any version as required.

User interface impact: Specifying a non-default handler can result in a different user interface and workflow during signing.

Seed values for specifying handlers and signature types are the following:

- **filter**: `filter` is the internal name of a signature handler. Signature handlers perform a number of functions including signature validation. While Acrobat ships with a default handler (Adobe.PPKLite), custom or third-party handlers such as those from Entrust and VeriSign may be used. The Acrobat SDK describes how to write a custom handler (Adbe.DocSign).

  Tip: `filter` is often used in conjunction with `version` when a minimum filter version is required.

- **subfilter**: `subfilter` is the internal name of the signature format, such as adbe.pkcs7.detached intended to be verifiable by signature handlers other than the one that created it. Signature handlers need to be able to understand the signature type (or format).

  Tip: Since it is possible that different handlers might be used for signing and validating, `filter` and `subfilter` are used together to assure that signing workflows with different components are interoperable. These properties are identical to those in the signature dictionary. For more information, refer to the ***PDF Reference***.

To specify a signature handlers and format type:

1. Create a signature field with an intuitive name and tooltip.
2. Create the JavaScript that gets the field object and uses the seed value method (Example 6.6).
3. Specify a `filter`.
4. If `filter` is specified, you may use the optional `version` as follows:
   - PPKLite for Acrobat 6.X: 0
   - PPKLite for Acrobat 7.x: 1
   - Custom handlers: Any.
5. Enter the handler name and `subfilter` type. Third parties may define their own subfilters but should follow the naming convention recommended in the ***PDF Reference***. The ***PDF Reference*** defines the following standard `subfilter` values:
   - adbe.x509.rsa_sha1
   - adbe.pkcs7.detached

- adbe.pkcs7.sha1

6. Run the JavaScript, save the document, and test the field.

Example 6.6: Seed value: Specifying signature components

```
// Obtain the signature field object:
var f = this.getField("mySigFieldName");

f.signatureSetSeedValue(
 {
filter: "Entrust.PPKEF",
subfilter: "adbe.x509.rsa_sha1
 } )
```

6.8 Specifying a Signature Hash Algorithm

When a signer's digital ID contains RSA public and private keys, it is possible to specify alternative signature hash algorithms. The default algorithm is SHA1, and the alternatives are listed in Table 5.

User interface impact: Once a document is signed, the signature's hash algorithm can be viewed by right clicking on a signature, choosing **Show Signature Properties**, and displaying the Document tab. The algorithm is displayed in the Hash Algorithm field.

Caution: If a signer may be using FIPS mode, do NOT specify MD5 or RIPMD160.

To specify a non-default algorithm:

1. Create a signature field with an intuitive name and tooltip.
2. Create the JavaScript that gets the field object and uses the seed value method (Example 6.7).
3. Specify the `digestMethod`. This can be an array of comma-separated items such as `['RIPEMD160','SHA384']`.
4. Run the JavaScript, save the document, and test the field.

Example 6.7: Hash algorithm seed value

```
// Obtain the signature field object:
var f = this.getField("mySigFieldName");

f.signatureSetSeedValue({
    digestMethod:['SHA384']
  });
```

6.9 Embedding Revocation Information in a Signature

Users (signers) have the option to embed certificate revocation status in a signature by turning on **Include signature's revocation status when signing** in their preferences. However, the default value is false (revocation information is not embedded), and document authors may need to force embedding of revocation information regardless of the users application settings. Embedding the signing certificate's revocation status in a document allows recipients to validate certificates (signatures) while offline and speeds up the revocation checking process. Moreover, if a certificate is revoked or expired at some time after signing, embedded revocation information enables the application to determine if a certificate was valid at the time of signing so that the signature status will remain valid.

Note: Only relevant if subFilter is adbe.pkcs7.detached or adbe.pkcs7.sha1. If the subFilter is adbe.x509.rsa_sha1 and adding revocation information is required, the signing operation fails.

To force embedding of certificate revocation information in a signature:

1. Create a signature field with an intuitive name and tooltip.
2. Create the JavaScript that gets the field object and uses the seed value method (Example 6.7).
3. Set `shouldAddRevInfo` to true.
4. Run the JavaScript, save the document, and test the field.

Example 6.8: Hash algorithm seed value

```
// Obtain the signature field object:
var f = this.getField("mySigFieldName");

f.signatureSetSeedValue({
    shouldAddRevInfo: true
  });
```

6.10 Specifying Certificate Properties for Signing

Certificate seed values are commonly used to restrict signing to particular certificates such as those issued by particular certificate authorities or containing numbers that specify certain policies with "object identifiers" or "OIDs." Authors specify which certificate signers must use by setting the certSpec object's properties (Table 8). These can be preferences or requirements. If a certificate cannot be found that matches a required certificate seed value, a URL can be provided to allow the signer to get more information such as how to obtain an appropriate certificate.

Certificate specification can be used to streamline workflows. When one certificate is allowed, the digital ID dialog is bypassed and the signer is directed to sign and save immediately. Signing fails if the selected certificate is not an exact match. It is also often expedient to provide a URL value so that users are directed to a help page or some location where a digital ID can be obtained.

Figure 57 Seed value: Specifying certificates for signing

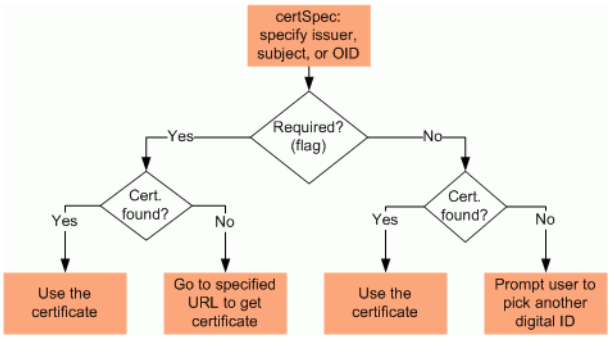


Table 8 Seed values: certSpec properties

Property	Type	Description
flags	number	A set of bit flags controlling which of the following properties of this object are required. The value is the logical OR of the following values, which are set if the corresponding property is required: **1**: subject **2**: issuer **4**: oid **8**: subjectDN (Acrobat 8 and later) **16**: issuerDN (Acrobat 8 and later) **32**: keyUsage (Acrobat 8 and later) **64**: url (Acrobat 8 and later) If this field is not present, all properties are optional. Usage: 1 specifies `subject`, 3 specifies `subject` and `issuer`, and 6 specifies `issuer` and `oid`. That is, values can be added. If this field is not present, all properties are optional.
issuer	array of certificate objects	One or more issuers that are acceptable for signing. The issuer can be a root or intermediate root certificate. Access to the physical, DER-encoded certificate is required. It is identified by a path to a discrete file in the format of [`"/c/test/root.cer"`].

Table 8 Seed values: certSpec properties

Property	Type	Description
keyUsage	array of integers	(Acrobat 8.0) Integers in HEX or decimal that specify the **`keyUsage`** extension that must be present in the signing certificate. Each integer is constructed as follows: There are two bits used for each keyUsage type (defined in RFC 3280) starting from the least significant bit: digitalSignature(bits 2,1) nonRepudiation(4,3) keyEncipherment(6,5) dataEncipherment(8,7) keyAgreement(10,9) keyCertSign(12,11) cRLSign(14,13) encipherOnly(16,15) decipherOnly(18,17) The value of the two bits have the following semantics: **00**: The corresponding keyUsage is not allowed. **01**: The corresponding keyUsage is required. **10 and 11**: The state of the corresponding keyUsage doesn't matter. For example, if it's required that keyUsage must require **`digitalSignature`** and the state of all other's doesn't matter, then the corresponding integer would be 0x7FFFFFFD. That is, to represent digitalSignature, set 01 for bits 2 and 1 respectively, and set 11 for all other keyUsage types.
oid	array of strings	One or more policy OIDs that must be present in the signing certificate's policy. The OID is part of the value of the certificate's certificate policy field. This property is only applicable if the issuer property is present. `oid` and `issuer` can be used together to specify a certificate that has the selected policy.
subject	array of certificate objects	One or more subjects that are acceptable for signing. The subject property identifies specific individuals (as certificate owners) that can sign. Access to the physical, DER-encoded certificate is required. It is identified by a path to a discrete file in the format of [`"/c/test/root.cer"`].
subjectDN	array of certificate objects	(Acrobat 8.0) Each object specifies a subject distinguished name (DN) acceptable for signing. More than one DN may be specified, but a signing certificate must satisfy at least one of the DNs by containing all the attributes specified in the matching DN. DN attribute restrictions are specified by adding them as properties. The properties' key names can either be the corresponding attributes' friendly names or OIDs (as defined in RFC 3280). The properties' value must be of type string. For more information about the various attributes and their types, refer to RFC 3280.
url	string	A URL that can be used to enroll for a new certificate if a matching one is not found, such as **`https://aardvark.corp.example.com/`**. Works in conjunction with **`urlType`** (if present). A degenerate use of this property is when the URL points to a Web service that is a digital ID store such as a roaming ID server. In that case, the URL indicates that as long as the signer has a digital ID from that Web service, it is acceptable for signing.

Table 8 Seed values: certSpec properties

Property	Type	Description
urlType	string	(Acrobat 8.0) The `url` type. If this attribute isn't present, it's assumed that the `url` points to a HTML site. There are two supported types: **HTML**: An HTML website. Acrobat uses the Web browser to display its contents. **ASSP**: A URL to a web service using the ASSP protocol for roaming ID servers.

6.10.1 Specifying Signing Certificates Origin

Authors can limit potential signers to individuals or groups as follows:

- `subject` limits potential signers to only those specified individuals. Signers could be limited to one or more people.
- `issuer` limits signers to those with certificates that chain up to a common, shared issuer. For example, all of a company's employees may use the company's certificate as an intermediate certificate and that certificate could be used as the issuer.
- `subjectDN` limits signers to those with certificates that match all the attributes of one of the listed DNs. For example:
  - `{cn:"Alice", ou:"Engineering", o:"Acme Inc"}`. For details about the friendly names of DN attributes (cn, o, ou, and so on), refer to the RDN Object in the *JavaScript for Acrobat API Reference*.
  - `{cn:"Joe Smith", ou:"Engineering", 2.5.4.43:"JS"}`, where OID 2.5.4.43 is used to carry out matching for the "Initials" attribute.

The following is sample code to define the above DN:

```
var subjectDN = {cn:"Joe Smith", ou:"Engineering"};
subjectDN["2.4.5.43"] = "JS";
```

Attributes whose value is of type `DirectoryString` or `IA5String` can be specified as shown in the example above, whereas all other value types, e.g. dateOfBirth whose value is of type GeneralizedTime, the value needs to be specified as a hex encoded binary string.

To specify a certificate:

1. Create a signature field with an intuitive name and tooltip.
2. Get the required certificates and install them in some accessible location.

   Tip: They must be in a `.cer` files in a DER format.

3. Create the JavaScript that gets the field object and uses the seed value method. Use `security.importFromFile` to get the DER- encoded certificates from their installed location (Example 6.9).
4. Add the `subject` and `issuer` properties to the `certspec` object.
5. Enter a flag value to indicate whether the value is required or not. Either or both the `subject` and `issuer` may be required.
6. Run the JavaScript, save the document, and test the field.

Example 6.9: Certificate issuer and subject seed value

```
// Obtain the signature field object:
var f = this.getField("mySigFieldName");

var mySubjectCert = security.importFromFile("Certificate", "/C/Temp/
nebwhifflesnit_DER.cer");
var myIssuerCert = security.importFromFile("Certificate", "/C/Temp/
nebsCompany_DER.cer");

f.signatureSetSeedValue(
 {
   certspec: {
     subject: [mySubjectCert],
     issuer: [myIssuerCert],
     flags: 3
   }
 } )
```

6.10.2 Specifying Certificates by Key Usage

Acrobat's default signature handler allows signing with certificates where the **Key usage** field is *Sign transaction* or *Sign document*. However, the **keyUsage** seed value allows you to override the default behavior and limit signing to those certificates where the keyUsage is set to any value defined in RFC 3280 (see Table 8). While the seed value could be used to require or disallow any of RFC 3280 **keyUsage** values, the two most common cases allow or disallow digitalSignature(bits 2,1) (displayed as *Sign transaction* in Acrobat's Certificate Viewer) or nonRepudiation(4,3) (displayed as *Sign document* in Acrobat's Certificate Viewer). However, any combination of uses may be set.

To restrict signing to a certificate with a particular **keyUsage**:

1. Create a signature field with an intuitive name and tooltip.
2. Create the JavaScript that gets the field object and uses the seed value method (Example 6.10).
3. Specify the keyUsage value in HEX:
    1. Specify 00, 01, 10, or 11 for each of the keyUsage values beginning with the least significant bit (the last one in the list in Table 8). For example:
        - digitalSignature is disallowed and non repudiation is required, and other values don't matter: 11111111111110100. Convert to HEX: 3FFF4
        - digitalSignature is required and non repudiation is disallowed, and other values don't matter: 11111111111110001. Convert to HEX: 3FFF1
    2. Remove the 3 and prepend the HEX value with 0x7FFF so it is in the correct HEX 32-bit format such as **0x7FFFFFF1**.
4. Enter a flag value to indicate whether the value is required or not. Set 32 if **keyUsage** is required and there are no other **certspec** properties.
5. Run the JavaScript, save the document, and test the field.

Example 6.10: Certificate key usage seed value

```
// Obtain the signature field object:
var f = this.getField("mySigFieldName");

f.signatureSetSeedValue({
  certspec: {
    keyUsage: [0x7FFFFFF1], //Set KeyUsage to "digitalSignature"
    flags: 32 //Require keyUsage
    },
  });
```

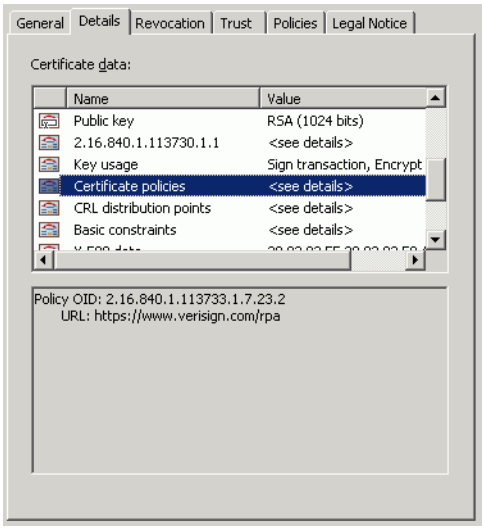
6.10.3 Specifying Certificates by Policy

For legal reasons, policies are often associated with certificates. One way policies are identified is through an object identifier (OID), a unique series of numbers in the certificate policies' field that identifies the policy. Since an **oid** is always used with the **issuer**, authors can use this seed value pair when a company issues different certificates with different policies and it is necessary to restrict signing to certificates associated with a certain policy.

To restrict signing to a certificate containing a specific policy:

1. Create a signature field with an intuitive name and tooltip.
2. Create the JavaScript that gets the field object and uses the seed value method (Example 6.11).
3. Specify the **issuer**.
4. Specify the **oid**. A policy OID is part of the value of the certificate's certificate policy field (Figure 58).
5. Enter a flag value to indicate whether the value is required or not. A value of 6 is recommended since **issuer** and **oid** must be specified together.
6. Run the JavaScript, save the document, and test the field.

Figure 58 Policy OID



Example 6.11: Certificate policy seed value

```
var myIssuerCert = security.importFromFile("Certificate", "/C/Temp/
nebsCompany_DER.cer");

// Obtain the signature field object:
var f = this.getField("mySigFieldName");

f.signatureSetSeedValue(
 {
   certspec: {
     issuer: [myIssuerCert],
     oid: ["2.16.840.1.1.113733.1.7.23.2"],
     flags: 6
   }
 } )
```

6.10.4 Specifying a URL When a Valid Certificate is not Found

When a valid certificate is not found, users can be redirected to a URL during the signing workflow. The URL may be to a server with a certificate repository; or, more likely, the URL may be a link to a Web page describing how to obtain a new or valid certificate.

To specify a certificate with a specific policy:

1. Create a signature field with an intuitive name and tooltip.
2. Create the JavaScript that gets the field object and uses the seed value method (Example 6.12).
3. Specify a certificate as described in one of the previous sections. Use `issuer` and/or `subject`.
4. Specify the URL. The URL can point to a certificate server or to instructions for getting a certificate.
5. Run the JavaScript, save the document, and test the field.

Example 6.12: Alternate certificate URL seed value

```
// Obtain the signature field object:
var f = this.getField("mySigFieldName");

var mySubjectCert = security.importFromFile("Certificate", "/C/Temp/
nebwhifflesnit_DER.cer");

f.signatureSetSeedValue(
 {
   certspec: {
     subject: [mySubjectCert],
     url: "https://aardvark.corp.example.com/",
   }
 } )
```

6.10.5 Restricting Signing to a Roaming ID

Fields can be required to be signed with roaming IDs by specifying the **`certspec`** **`url`** and **`urlType`** properties. By providing the roaming ID server URL and the ASSP protocol as arguments, only roaming

IDs associated with the specified server will appear in the signing dialog's digital ID drop-down list when a user attempts to sign the field.

To require signing only with a roaming ID:

1. Create a signature field with an intuitive name and tooltip.
2. Create the JavaScript that gets the field object and uses the seed value method (Example 6.13).
3. Specify the roaming ID server URL.
4. Specify `ASSP` as the URL type.
5. Run the JavaScript, save the document, and test the field.

Example 6.13: Roaming ID seed value

```
// Obtain the signature field object:
var f = this.getField("mySigFieldName");

f.signatureSetSeedValue(
 {
certspec: {
     url: "https://myroamingIDserver.arcot.com:9777",
     urlType: "ASSP",
   }
 } )
```

6.11 Custom Workflows and Beyond

Advanced document and workflow customization is beyond the scope of this document. However, keep in mind that Acrobat's security APIs allow users many opportunities for customization. Document developers can easily create custom signing menu items, automate tasks, and perform other operations beyond those described in the preceding seed value sections.

For example, Example 6.14 performs a number of operations that would simplify signing operations in an enterprise setting. The script adds a **Request Employee Signature** to the toolbar and set up a number of automatic actions. When a user selects the menu item, a signature field with predefined properties is automatically created in the needed document location, and the field's seed values are set.

Note: For more information, refer to the online *Acrobat JavaScript Scripting Guide*, *JavaScript for Acrobat API Reference*, *PDF Reference*, and the Acrobat SDK.

Example 6.14: Automating signing tasks

```
//**************************************************************************
//File: seedValue.js
//Purpose: Demo how to set certificate constrictions into a signature field
//Steps: 1. Add a menu item under Tools,called Request Employee Signature
//       2. Add a signature and text field(for display)to the current open file
//       3. Set seed value
//          3.1 Wrap certificate object
//          3.2 Set seed value to the added signature field
```

```
//                reason: "I am approving this document"
//                certSpec:
//                     issuer: Example/MyCompanyCA (the root)
//                     oid: 2.5.29.16 the oid of Example/MyCompanyCA
//                     url: https://my.corp.example.com/
//                    flag: 2 set limits on issuer
//       4. Display seed value added to the sig field to the added text field.
//*********************************************************************
// 1. Add a Tools menu item called Request Employee Signature
app.addMenuItem
({
 cName: "Request Employee Signature",
 cParent: "Tools",
 cExec: "setSeedValues()",
 cEnable: "event.rc = (event.target != null);",
 nPos: 0
 });

//Run function when menu item "Request Employee Signature" is clicked
function setSeedValues(){
//modify the following according needs
   var sigfieldName = "aSigField";
   var myReasons = ["I am approving this document"];
   var myIssuer;
   var oids = ["2.5.29.16"];
   var url = "https://seneca.corp.example.com/";
   var certSpecFlag = 2;//constricts on issuer, 6: issuer + OID, 1: users to
sign, 7: issuer + oid + user
   var svFlag = 0;      //no restrictions

   try{
       //2. add a sig field called "aSigField" and a text field
      var field = this.addField(sigfieldName,"signature",0,[180,640,352,680]);/
/1st page
      field.borderStyle = border.s;
      field.fillColor = color.ltGray;
//a text field to display what seed values set to the sig field
        var textField = this.addField("aText", "text",0, [110,360,500,550]);
        textField.borderStyle = border.s;
        textField.fillColor = color.yellow;
        textField.multiline = true;
        textField.display = display.hidden;  //hiden form screen and print
        textField.setAction("MouseUp", "event.target.display = display.
hidden;");//click field, field disappears

      //3. set seed value
      //3.1 set  up issuer's certificate object
      var myissuerDN = {CN:"Enterprise Services CA", OU:"VeriSign Trust
Network", O:"Example Systems Incorporated"};
      var mykeyUsage = ["kDigitalSignature","kCRLSign"];
      var myMD5Hash = "BF70 913F F8D6 D60A 47FE 8253, 3081 5DB4";
      var mySHA1Hash = "6b e8 46 06 39 f5 65 18 48 b2 f8 3a b1 46 3f 56 02 be 06
c3";
      var myserialNumber = "3e 1c bd 28";
```

```
     var mysubjectCN = "Example Root CA";
     var mysubjectDN = {CN:"Example Root CA",OU: "Example Trust Services",O:
"Example Systems Incorporated",C:"US"};
     var myusage = {endUserSigning:true};
       var ExampleRootCertBinary =
"308204A130820389A00302010202043E1CBD28300D06092A864886F70D01010505003069310
//-----------<snip>----------
440512D9E9B47DB42A57C1FC2A648B0D7BE92694DA4F62957C5781118DC8751CA13B2629D4F2B3
2BD31A5C1FA52AB0588C8";

//var myIssuer = {binary:ExampleRootCertBinary,issuerDN:myissuerDN,keyUsage:
mykeyUsage,MD5Hash:myMD5Hash,
//SHA1Hash:mySHA1Hash,serialNumber:myserialNumber,subjectCN:
mysubjectCN,subjectDN:mysubjectDN,usage:myusage};
     var myIssuer = security.importFromFile("Certificate", "/c/test/root.
cer");//if import from an external reference
     // 3.2 set up seed value
       field.signatureSetSeedValue({reasons:myReasons,certspec:{ issuer:
[myIssuer],/*oid:oids,*/url:url,flags:certSpecFlag}, flags:svFlag});
     //4. Display seed value added to the signature field to the new text field
     var result = "";
     var w = field.signatureGetSeedValue();
     for(i in w)
       result += ( i + " = " + eval("w." +i) + "\n");

     var z = w.certspec;
     for(i in z)
       result += ( i + " = " + eval("z." +i) + "\n");

       textField.value = result + "** Click on me to make me disappear **";
       textField.display = display.show; //display what seed values were set
   }catch (e){
     app.alert("setSeedValues(): " + e );
   }
}
```

Appendices

Quick Keys

What's Changed Across Releases

Supported Standards

Etc.

A Changes Across Releases

For a version by version listing of other changes, see the following:

- For 10.0 and later, see the introduction in this guide.
- Prior to 10.0, refer to the *Digital Signatures Addendum*.

The Acrobat family of products introduces new features and enhancements with each release. The following technical details list only a fraction of the changes that occur with each new product release:

- Digital ID Related Files and Storage Mechanisms
- Algorithms for Digital Signatures
- Seed Values
- Signature Appearances
- XML Form Support for Signatures
- Directory Servers
- FDF (Data Exchange) Files

A.1 Digital ID Related Files and Storage Mechanisms

A digital ID may reside in a local or networked file or on some external hardware. There are several digital ID-related file types and storage mechanisms a user might encounter in signing and certificate encryption processes (Table 1). Digital IDs (both the certificate with the private key) and certificates (with a public key but no private key) are provided to Acrobat via digital ID service providers (one type of "cryptographic service provider" or CSPs).

In many cases, the digital ID is stored on a local or networked file. Common file locations include the Windows Certificate Store (where they can be used both by Acrobat and other Windows applications) and the default Acrobat cache (used only by the Acrobat family of products). Others IDs may reside on external PKCS#11 hardware that is plugged into the computer or on a roaming ID server. Regardless of location the application remembers the location when the ID is registered (imported) and is added to the Security Settings Console's digital ID list.

The Acrobat family of products supports the following digital ID providers:

- **PKCS#12 files**: A common file format that contains the entire digital ID. It is used by Acrobat on Windows as well as Macintosh.
- **Windows Certificate Store**: A local store that can import and export various file formats and that can be used by both Windows MSCAPI-compliant programs including Acrobat products.
- **PKCS#11 devices**: External devices that store digital ID data. Users do not interact with a file.
- **Roaming ID servers**: The private key is known only to a remote server. The server sends the certificate and its public key to users after they authenticate. Users can import and export the certificate from Acrobat.
- **APF files**: A legacy format that is no longer used.

Note: There are no changes to this feature for 10.x.

Table 1 Digital ID-related file types

Type	Description	5.x	6.x	7.x	8.x	9.x
.acrobat security	An XML format encapsulated in a PDF which stores security settings for import and export. **Contains**: Digital ID (public and private keys)					Export Import
PKCS#12: .pfx (Win), .p12 (Mac)	**Personal Information Exchange Syntax Standard**: Specifies a portable, password protected, and encrypted format for storing or transporting certificates. **Contains**: Digital ID (public and private keys)		Export Import	Export Import	Export Import	Export Import
.fdf	An Adobe file data exchange format used for importing and exporting settings and certificates (usually PKCS#12 files).	Export Import	Export Import	Export Import	Export Import	Export Import
PKCS#7: .p7b, .p7c	**Certificate Message Syntax (CMS)**: Files with .p7b and .p7c extensions are registered by the Windows OS. Acrobat products can import and export these files. **Contains**: Certificate and public key only		Export Import	Export Import	Export Import	Export Import
.cer	**Certificate format**: A Microsoft format for digital IDs usually stored in the Windows Certificate Store. **Contains**: Certificate and public key only		Export Import	Export Import	Export Import	Export Import
.apf	**Adobe Profile Files (Legacy)**: Not used after Acrobat 5. Files can be upgraded by double clicking them. **Contains**: Digital ID (public and private keys)	Import Export	Import	Import	Import	n/a

APF files

The ability to create an APF file (Acrobat Personal File) only existed in Acrobat 4 and 5. The file was a container for an Acrobat generated, self-signed digital ID plus an address book that held other peoples public keys for the purpose of applying Certificate Security (for signature validation).

Beginning with Acrobat 6, the ability to create an APF file was dropped because P12/PFX usage was introduced along with a separate address book for managing other peoples certificates. Acrobat 6 would import the certificates from an APF file and allow export of a digital ID to a P12 file. Acrobat 7 would still open the APF file and allow access to its contents, it no longer offered the ability to export your private key. Therefore, if you want to convert your APF file to a P12 file you can only use Acrobat 6.

Acrobat 9 does not support APF in any way, so if you need to use or access an APF file after Acrobat 9 is installed, you will have to uninstall A9, reinstall A6, import and then export the digital ID from the APF to a P12 file, and then reinstall A9.

A.2 Algorithms for Digital Signatures

Applying a digital signature involves the use of two algorithms: one for creating the message digest (a document hash) and one for encrypting the digest:

- **Message digest**: The default algorithm used to digest signed data becomes stronger across releases. Specifying an alternate algorithm is possible through registry configuration.
- **Encryption algorithm**: That algorithm is selected based on the signing digital ID's private key which specifies an algorithm and key length when the public-private key pair is generated.
- **Libraries**: The statically linked libraries (not dlls) in use are:
  - Acrobat 9: CryptoC 6.1.1 (Mac), CryptoC 6.3.2, CryptoC-ME 2.1 (Windows)
  - Acrobat 10: CryptoC 6.1.1 (Mac), CryptoC 6.4.0, CryptoC-ME 3.0 (Windows)

Figure 1 Algorithms used for signing

Message digest creation algorithms

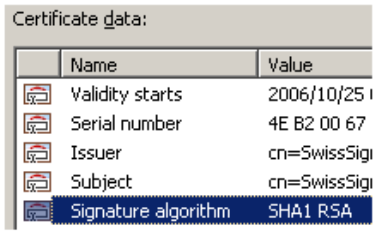
Specifed by aSignHash or tSignHash (Defaults in Red)

	SubFilter value specified by aSignFormat, seed value, JavaScript, or PubSec handler				
V.	**adbe.pkcs7.detached**	**adbe.pkcs7.sha1**	**adbe.x509.rsa.sha1**	**ETSI.CAdES.detached**	**PDF #**
10.0	MD5, SHA1, **SHA256**, SHA384, SHA512, RIPEMD160	SHA1	Same as adbe.pkcs7.detached	Same as adbe.pkcs7.detached	v.1.7
9.0	MD5, **SHA1**, **SHA256 (9.1)**, SHA384, SHA512, RIPEMD160			N/A	v.1.7
8.0	MD5, **SHA1**, SHA256, SHA384, SHA512, RIPEMD160				v.1.7
7.0	MD5, **SHA1**, SHA256				v.1.6
6.0	MD5, SHA1				v.1.3
4-5.0	MD5, SHA1				v.1.3

Message digest encryption algorithms

Specified by the signing digital ID's (determines algorithm and key length when the public-private key pair is generated

V.	Encryption algorithms	PDF #	Digest creation compatibility
8.0-10.0	RSA and DSA up to 4096-bit	v.1.7	RSA supports all algorithms and signataure types (subFilter values). DSA only supports SHA1 and adbe.pkcs7.detached.
7.0	RSA and DSA up to 4096-bit	v.1.6	
6.0	RSA up to 4096 bit	v.1.5	
4-5.0	RSA up to 1024 bit	v.1.3	



A.3 Seed Values

Signature field seed values enable document authors to limit a signer's choices when signing a particular signature field. As shown in Table 2, document authors can create documents with behaviors and features that meet specific business needs, thereby enabling administrative control of signature workflows.

For more information, refer to the following (Go to http://www.adobe.com/go/acrobat_security):

- *Developing Acrobat Applications with JavaScript*

- *JavaScript for Acrobat® API Reference*

Table 2 Seed values: Changes across releases

Seed value	First support for seed value	6	7	8	9	10
certspec	Specifies that certain certificates must be used for a particular signature field. **6.0-7.x**: Supports subject, issuer, and oid. **8.x**: Adds support for subjectDN, issuerDN, keyUsage, url, and urlType	X	X	X	X	X
filter	The language-independent name of the security handler to be used when signing.	X	X	X	X	X
flags	A set of bit flags controlling which properties are required. **6.0-7.x**: 1: filter, 2: subFilter, 4: version, and 8: reasons. **8.0**: 16: legalAttestations, 32: shouldAddRevInfo, and 64: digestMethod.	X	X	X	X	X
legalAttestations	A list of legal attestations that the user can use when creating an MDP (certification) signature.	X	X	X	X	X
mdp	Can be used to force a certification signature as well as to control permitted document changes.	X	X	X	X	X
reasons	A list of reasons that the user is allowed to use when signing. **8.0**: Supports disabling signing reasons.	X	X	X	X	X
subFilter	An array of acceptable signature formats.	X	X	X	X	X
timeStampspec	Specifies a timestamp server using the `url` and `flags` properties.	X	X	X	X	X
version	The signature handler version to be used to sign the signature field. Valid values are 1 and 2. **8.0**: Must be set to 2 if this seed value object contains Acrobat 8-specific content marked as required.	X	X	X	X	X
digestMethod	The algorithm used to created the message digest. **6.0-7.x**: MD5, SHA1. **8.0**: Adds support for SHA256, SHA384, SHA512, and RIPEMD160.		X	X	X	X
shouldAddRevInfo	Controls how the application does certificate and chain revocation checking.			X	X	X
lockDocument	Allows the author to add a Lock Document checkbox to the signing dialog so a signer can lock the document at the time of signing.				X	X
AppearanceFilter	A text string naming the appearance required to be used when signing the signature field.				X	X

A.4 Signature Appearances

The format has always been PDF, but appearance file names and behavior have evolved across releases.

Table 3 Changes in signature appearances

V.	Filename	Behavior
10.0	No change	No change.
9.0	No change	Signature status icon no longer appears with the appearance.
8.0	No change	No change.
7.x:	No change	No change.
6.0:	appearance.acrodata	One signature appearance file per OS login. Each page in the PDF corresponds to a signature appearance. Each page also contains private data that is hung off of the Page object. This data contains the settings for checkboxes and other options in the Configure Signature Appearance dialog.
5.05:	<apf basename>.pdf	One appearance file per apf file. 5.0 appearance files are forward compatible with 6.0.

A.5 XML Form Support for Signatures

Support for security features in dynamic XML forms authored with Designer has improved across versions. With version 8.0, there is full support for the following:

- **Approval signatures**: Sometimes referred to as signing.
- **Certification signatures**: Sometimes referred to as certifying.
- **MDP+**: modification-detection-protection (field locking).
- **Reader extensions**: Enabling usage rights so Reader users can sign documents with existing fields.

Table 4 Signature type support in XML forms

	Static XML forms					Dynamic XML forms				
Signature Type	**6.x**	**7.x**	**8.x**	**9.x**	**10.x**	**6.x**	**7.x**	**8.x**	**9.x**	**10.x**
Approval	Yes	Yes	Yes	Yes	Yes		Yes	Yes	Yes	Yes
Certification (DocMDP)	Yes	Yes	Yes	Yes	Yes		Yes	Yes	Yes	Yes
Field protection (FieldMDP)			Yes	Yes	Yes			Yes	Yes	Yes
Reader extensions (UR3)	Yes	Yes	Yes	Yes	Yes		Yes	Yes	Yes	Yes
XML data signature		Yes	Yes	Yes	Yes		Yes	Yes	Yes	Yes

A.6 Directory Servers

Acrobat products ship with preconfigured directory servers. The servers are used by the Trusted Identity Manager to locate certificates for import. Users can trust the imported certificates for signing and

certifying documents as well as for encrypting documents prior to sending them to the certificate owner.

Table 5 LDAP certificate repositories

Version	Default Directory Servers
7.x	VeriSign Internet Directory Service GeoTrust Directory Service IDtree Directory Service
8.0	VeriSign Internet Directory Service
9.0 and later	VeriSign Internet Directory Service

A.7 FDF (Data Exchange) Files

The FDF format and .fdf files enable the import and export of certificate data and security settings. End users can share application settings, and administrators can distribute trust anchors and server settings across their organization. Acrobat products can create the files as part of the export process, or files can be programmatically generated by a custom program or script that writes a file conforming to the FDF Data Exchange Specification.

Tip: With 9.0, FDF is being deprecated and replaced with the security setting import/export feature.

Table 6 FDF changes across releases

First support for FDF feature	5	6	7	8	9-10
Import and export of Acrobat-generated self sign certificates.	X	X	X	X	X
FDF file signing for origin verification.		X	X	X	X
Import and export of any supported certificate type (including those in the Windows store).		X	X	X	X
Import and export of directory server settings.		X	X	X	X
Import and export of Adobe LifeCycle Rights Management Server settings.			X	X	X
Import and export of timestamp server settings.			X	X	X
Import and export of roaming ID server settings.				X	X

B Supported Standards and RFCs

Acrobat's features adhere to widely accepted standards as listed in Table 7.

Table 7 Standards support

Reference	Feature
PDF Reference 1.7 (ISO 32000-1) http://www.adobe.com/devnet/pdf/pdf_reference.html. See also PDF for Archive (PDF/A) and PDF for Exchange (PDF/X) at http://www.iso.org.	Representing signatures in the PDF language.
RFC 3280, Internet X.509v3 Public Key Infrastructure, Certificate and Certificate Revocation List (CRL) Profile. http://www.ietf.org	CRL revocation checking, chain building, path validation, cross certificates, multiple chains.
RFC 2560, X.509 Internet PKI Online Certificate Status Protocol-OCSP. http://www.ietf.org	OCSP revocation checking.
RFC 3161, Internet X.509 Public Key Infrastructure Time-Stamp Protocol http://www.ietf.org	Timestamping: signing and signature validation.
RFC 3281, Attribute Certificate Profile, S. Farrell, R.Housley April 2002. http://www.ietf.org	Attribute certificates.
RFC 2437, PKCS #1: RSA Cryptography Specifications Version 2.0 (1024, 2048, 4096). http://www.ietf.org	A format used for creating a digital signature object which is embedded in a document.
RFC 2898, PKCS #5: Password-Based Cryptography Specification Ver. 2.0. http://www.ietf.org	Password security.
RFC 2315, PKCS #7: Cryptographic Message Syntax, Version 1.5. http://www.ietf.org	A format used for creating a digital signature object which is embedded in a document.
PKCS #11: URI Scheme http://www.ietf.org	Cryptographic token interface (smart cards, tokens, etc.)
RFC 1321, The MD5 Message-Digest Algorithm http://www.ietf.org	Creating a document hash during signing.
RFC 3174, US Secure Hash Algorithm 1 (SHA1) http://www.ietf.org	Creating a document hash during signing.
FIPS PUB 186-2, Digital Signature Standard, describes DSA signatures. http://csrc.nist.gov/publications/fips/fips186-2/fips186-2-change1.pdf	Digital signatures.
FIPS PUB 197, Advanced Encryption Standard (AES 128, 256). http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf	Certificate security.
ISIS-MTT Specification v.1.1 March 2004. http://www.isis-mtt.org/index.php?id=460&L=1	Attribute certificates.
NIST PKITS "Public Key Interoperability Public Key Interoperability Test Suite Certification Path Validation"	Chain building and path validation, including cross certificates and multiple chains.
OIDS. ASN.1	Object identifiers (OIDs)

Table 7 Standards support

Reference	Feature
RFC 2396, Uniform Resource Identifiers (URI): Generic Syntax. http://www.ietf.org	All.
RFC 2595, Using TLS with IMAP, POP3 and ACAP. http://www.ietf.org	The PLAIN authentication mechanism used by the roaming ID feature.
RFC 3778, The application/pdf Media Type. Adobe Systems Incorporated.	Describes PDF media type, digital signature and encryption
ETSI 102 778 PDF Advanced Electronic Signatures (PAdES), Parts 1,2,3 and 4. http://pda.etsi.org/pda/queryform.asp	Digital signature; especially LTV.
ETSI/ESI Technical Standard (TS) 102 778 http://pda.etsi.org/pda/queryform.asp	Digital signatures.
JITC: Joint Interoperability Test Command PKI compliance test suite	DoD-mandated PKI test suite. Compliant since 7.x. See http://blogs.adobe.com/security/tag/jitc.

Table 8 Support for APIs, organizations, etc.

MSCAPI	Microsoft's CryptoAPI
Keychain	Macintosh's CryptoAPI
Esign	A U.S. law that both Acrobat and EchoSign signatures conform to.

Glossary of Security Feature Terms

Table 9 provides a comprehensive list of security terms and acronyms used in the fields of digital signatures and document security.

Table 9 Security Terms

.apf	See Adobe Profile Files.
.cer	A Microsoft format for digital IDs often stored in the Windows Certificate Store. These IDs can be used by Windows programs as well as the Acrobat product family.
.p12	See PKCS#12.
.p7b	See PKCS#7.
.p7c	See PKCS#7.
.pfx	See PKCS#12.
AATL	See Adobe Approved Trust List
AC	See attribute certificate.
Acrobat's Public Key Infrastructure Library	A standalone PKI toolkit written in C++ with the intention of being completely portable and usable in different applications, including but not limited to, Acrobat and GUI-less servers. ASPKI supports RFC 3280 and NIST compliant chain building and path validation, including support for cross certificates and multiple chains; multiple revocation protocols like CRL (RFC3280) and OCSP (RFC2560); time stamping (RFC3161); and embedded revocation information along with a signature to achieve signature archival.
AdES	See advanced electronic signatures
Adobe Approved Trust List	An Adobe program designed to facilitate trust in PDF signatures by downloading a list of trusted, high assurance root and ICA certificates to Acrobat and Reader v9.0 and above.
Adobe LiveCycle Rights Management Server	Adobe LiveCycle Enterprise Suite (ES) is a SOA J2EE-based (Java 2 Enterprise Edition) server software product from Adobe Systems Incorporated used to build applications that automate a broad range of business processes for enterprises and government agencies.
Adobe Policy Server	As of Acrobat 9, Adobe Policy Server is renamed to Adobe LiveCycle Rights Management Server.
Adobe Profile Files	Adobe's legacy certificate format not used after Acrobat 5. The certificates are stored in . apf files. This format is not supported as of version 9.0.
advanced electronic signatures	A type of electronic signature described in the European Union Signature Directive. Differentiated from a Qualified Electronic Signature in that it may not use a QEC.
AIA	See authority information access.
AIIM	See Association for Information and Image Management.
ALCRMS	See Adobe LiveCycle Rights Management Server.
approval signature	A signature used to indicate approval of, or consent on, the document terms. Acrobat and Reader recognize both approval and certification signatures.
APS	See Adobe Policy Server.

Table 9 Security Terms

ASPKI	See Acrobat's Public Key Infrastructure Library.
Association for Information and Image Management	AIIM is a non profit community that provides education, research, and best practices to help organizations find, control, and optimize their information as well as understand the challenges associated with managing documents and business processes.
attribute certificate	A file that contains the supplemental attributes and extension that is bound to a PKC, but does not itself contain any key data.
authority information access	An extension that is part of a PKC which contains information on how to access either PKC's of issuing certificate(s) (least used) or where to access OCSP revocation information which is what this extension is primarily used for.
basic constraint	An extension within a public key certificate that defines whether or not the certificate has been issued to a CA.
CA	See certificate authority (CA).
CAdES	See Cryptographic Message System Advanced Electronic Signatures.
CAPI	See MSCAPI.
CDS	See Certified Document Services (CDS).
CDS digital ID	A digital ID issued by a Certified Document Services provider.
CDS digital ID certificate	See CDS digital ID.
CEN	European Committee for Standardization.
certificate	That part of a digital ID that contains the public key. Certificates are shared among participants of signature and certificate security workflows in order to verify participant identities.
certificate authority (CA)	An entity which issues digital certificates for use by other parties. It is an example of a trusted third party. CAs are characteristic of many PKI schemes.
certificate revocation list (CRL)	CRL is a method that public key infrastructures use to maintain access to cached or networked lists of unexpired but revoked certificates. The list specifies revoked certificates, the reasons for revocation (optional), and the certificate issue date and issuing entities. Each list contains a proposed date for the next release. Acrobat's CRL revocation checker adheres to RFC 3280 and NIST PKITS except for delta CRLs.
certification signature	A digital signature applied using an individual digital ID or organizational digital ID for the purpose of establishing the authenticity of a document and the integrity of a document's content, including its appearance and business logic.
certified document	A document to which a certification signature has been applied.
Certified Document Services (CDS)	An Adobe program where commercial CA's create a subordinate or ICA below that chains to the Adobe Root certificate. As the Adobe Root is automatically trusted by Acrobat and Reader v6.0 and above, signatures made with credentials that chain to it are also similarly trusted.
certify or certifying	The act of applying a certification signature to a document using the Acrobat "Certify" feature. Certifying helps establish document authenticity as well as the integrity of its content, including its appearance and business logic.
Click thru signature	A type of electronic signature where the signer is indicating their agreement with terms and indication to sign the document / process by clicking on a button, which might say "I accept." Typically, an audit log of the event is kept for evidentiary purposes and authentication may or may not be required. Sometimes a server applied digital (certification) signature may be applied after the click thru process to protect the integrity of the document.
CMS	See Cryptographic Message Syntax.

Table 9 Security Terms

CRL	See certificate revocation list (CRL).
Cryptographic Message Syntax	A syntax is used to digitally sign, digest, authenticate, or encrypt arbitrary message content which is described in RFC 3369.
Cryptographic Message System Advanced Electronic Signatures	An EU Advanced Signature Format relying on CMS signatures, as described in ETSI TS 101 733.
cryptographic service provider	Application software that allows it to use MSCAPI to communicate with cryptographic module APIs such as PKCS#11 modules, PFX files, and so on
CSP	See cryptographic service provider.
CSP	Certificate Service Provider. Alternate term for CA.
digest method	A hash algorithm used to create a one way hash of data. Acrobat supports six digest methods; MD5, SHA1, SHA256, SHA384, SHA 512, and RIPEMD 160.
digital ID	An electronic representation of data based on the ITU-T X.509 v3 standard, associated with a person or entity. It is often stored in a password-protected file on a computer or network, a USB token, a smart card, or other security hardware device. It can be used for digital signatures and certificate security. "Digital ID" is sometimes used interchangeably with "certificate"; however, a certificate is only one part of a digital ID which also contains a private key and other data.
digital signature	An electronic signature that can be used to verify the identity of the signer through the use of public key infrastructure (PKI) technology. Signers need a digital ID and an application capable of creating a signature.
digital signature algorithm	An encryption algorithm used to create a digital signature created by NIST as defined by FIPS 186.
digitally sign	To apply a digital signature using a digital ID.
document integrity	In signing workflows, document integrity refers to whether or not what was signed has changed after signing. That is, what the signer signed should be reproducible and viewable on the document recipient's end. For the document recipient to validate a signature, its important to determine to what document or what document version that signature applies. See message digest.
DSA	See digital signature algorithm.
EC	European Commission.
EE	See end entity certificate (EE).
EESSI	Electronic Exchange of Social Security Information.
eID	See electronic ID.
electronic ID	A broad term for any electronic ID. In the EU, it is commonly used to describe national-level identity cards.
electronic signature	Generic term. Generally defined as an electronic process which intrinsically links some tag (data, voice, image, key) to content that is being signed, is linked to the signer, and is generally capable of detecting changes in the document signed. A digital signature is a type of electronic signature, as are click thru and signature image.
embedded JavaScript	JavaScript that exists within a document rather than that which is executed from the JavaScript Console or through a batch process.

Table 9 Security Terms

embedded validation response	Information from the digital ID issuer that was used to apply the digital signature and that indicates if the digital ID was valid when the signature was applied. If the digital ID was valid and no one has tampered with the document, the signature will have a status of VALID. Once the digital ID expires or is revoked, it won't be possible to determine if the signature was valid at the time it was applied unless there is an embedded revocation response.
end entity certificate (EE)	The last element of a signing chain. By definition, an end entity certificate does not contain the basic constraint value CA, and is issued to an individual or a pseudo entity (e. g. a department or organization).
ETSI	European Telecommunications Standards Institute.
ETSI/ESI	ETSI/Electronic Signature & Infrastructure Technical Committee.
EU	European Union.
EU Experts Group	A generic term used widely within the EU. Of particular interest is the EUEG on Electronic Procedures.
EU Signature Directive	Directive 1999/93/EC on a Community Framework for Electronic Signatures. Established a Framework for Electronic Signatures and Standards in the European Union.
FIPS	Federal Information Processing Standards: These are publicly announced standards developed by the United States Federal government for use by all non military government agencies and by government contractors. Many FIPS standards are modified versions of standards used in the wider community (ANSI, IEEE, IOS, etc.).
FIPS 140	Federal Information Processing Standard 140: Standard which defines increasing levels of assurance for hardware and software based devices and applications for storing private keys. Level 1 is the lowest assurance, and level 4 is the highest, requiring devices to self-destruct and 'zeroize' themselves if they are compromised in any way.
hardware security module	While actually a generic term for any hardware device designed to securely store digital IDs, HSMs in common parlance are rack mounted servers or hardened PCI cards which are designed for higher security and higher volume cryptographic operations.
hardware token	A hardware device (typically a smart card or USB device) that contains the user's digital ID(s) and requires a password or other authentication method to access those IDs for the purpose of signing.
HSM	See hardware security module.
ICA	See intermediate certificate authority (ICA).
IDABC	Interoperable Delivery of European eGovernment Services to public Administrations Businesses and Citizens individual digital ID: A digital ID issued to an individual to digitally sign as them self (e.g. John Smith) as opposed to an organization or other non-human entity.
individual digital ID	A digital ID issued to individuals so that they can identify themselves during a digital signature process (e.g. John Smith) as opposed to an organization or other non-human entity.
intermediate certificate authority (ICA)	A type of CA characterized by the fact that the ICA's certificate may itself be signed by a different ICA, all the way up to a 'self-signed' root certificate. Certificates in between the end entity and root certificates are sometimes called "intermediate certificates" (ICAs) and are issued by the CA or ICAs underneath the CA.
ISO	International Standards Organization.
ITC	Information Communication Technologies.

Table 9 Security Terms

long term validation	The validation of a digital signature after certificate(s) associated with the signature has expired.
LTV	See long term validation.
message digest	Before Acrobat or Adobe Reader can verify if a document the signed version of the document has changed or not (has integrity), it must first have a way to uniquely identify what was signed. To do this, it uses a message digest. A message digest is a number which is created algorithmically from a file and which uniquely represents that file. If the file changes, the message digest changes. Sometimes referred to as a checksum or hash, a message digest is simply a unique number created at signing time that identifies what was signed and is then embedded in the signature and the document for later verification.
MS	Member State (one of the EU countries)
MSCAPI	Windows Microsoft Crypto API (MSCAPI) is the API that the application uses to access cryptographic service providers such as PFX files and PKCS#11 files. MSCAPI is also used by the application anytime it uses a Windows security feature.
National Institute of Standards	A non regulatory agency of the United States Department of Commerce's Technology Administration. The institute's mission is to promote U.S. innovation and industrial competitiveness by advancing measurement science, standards, and technology in ways that enhance economic security and improve quality of life.
NIST	See National Institute of Standards.
OCSP	See online certificate status Protocol (OCSP).
online certificate status Protocol (OCSP)	OCSP defines a protocol for determining the revocation status of a digital certificate without requiring a CRL. Unlike CRL, OCSP obviates the need to frequently download updates to keep certification status lists current. Acrobat's OCSP revocation checker adheres to RFC 2560.
organizational digital ID	A digital ID issued to an organization or non-human entity (for example, the Adobe Public Relations Department). It can be used by an authorized employee / process to perform signing operations, at the desktop or server, on behalf of the company.
PAdES	See PDF Advanced Electronic Signatures.
PDF Advanced Electronic Signatures	A five part standard (ETSI TS 102 778) which describes how to use the digital signature features of the Portable Document Format (PDF) to meet EU signature requirements.
PKC	See public key certificate.
PKCS	*A* group of Public Key Cryptography Standards authored by RSA Security
PKCS#11	Public key Cryptography Standard #11: A Public key cryptography Standard published by RSA Laboratories defining an API, called Cryptoki, to devices which hold cryptographic information and perform cryptographic functions.
PKCS#11 device	External hardware such as a smart card reader or token. It is driven by a module (a software driver such as a .dll file on Windows).
PKCS#11 digital ID	An ID on a PKCS11# device. A device may contain one or more IDs.
PKCS#11 format	Cryptographic Token Interface Standard: An encryption format used by smart cards, tokens, and other PKCS#11-compatible devices. The ID is stored on the device rather than on the user's computer.
PKCS#11 module	The software module that drives a PKCS#11 device.
PKCS#11 token	See PKCS#11 device.

Table 9 Security Terms

PKCS#12	Public Key Cryptography Standard #12: Personal Information Exchange Syntax Standard that specifies a portable, password protected, and encrypted format for storing or transporting certificates. The certificates are stored in .pfx (Windows) and .p12 (Macintosh) files. Unlike other formats, the file may contain private keys.
PKCS#7	Public Key Cryptography Standard #7: A Public key cryptography standard published by RSA Laboratories that defines the syntax/format for a digital signature. This format extends PKCS#1 information to include timestamps, digital certificates and more. Files with .p7b and .p7c extensions are registered by the Windows OS. If you double click on a . p7c file it will be viewed by a Windows application. Replaced by CMS.
PKCS#9	Public Key Cryptography Standard #9 (of #15 produced by RSA) Selected Object Classes and Attribute Types. Defines attributes that PKCS#7 uses.
PKI	See public key infrastructure.
point of single contact	The point of single contact for a member state.
Policy Server	As of Acrobat 9, Adobe Policy Server is renamed to Adobe LiveCycle Rights Management Server
Private key	The secret key in a PKI system, used to validate incoming messages and sign outgoing ones. A Private Key is always paired with its Public Key during those key generations.
privileged context	A context in which you have the right to do something that's normally restricted. Such a right (or privilege) could be granted by executing a method in a specific way (through the console or batch process), by some PDF property, or because the document was signed by someone you trust. For example, trusting a document certifier's certificate for executing JavaScript creates a privileged context which enables the JavaScript to run where it otherwise would not.
PSC	See point of single contact.
public key	The publicly available key in a PKI system, used to encrypt messages bound for its owner and to validate signatures made by its owner. A Public Key is always paired with its Private Key during those key generations.
public key certificate	A file that contains the numeric public key portion of a public/private key pair along with the associated extensions and attributes that are used to define who the certificate is for, what its validity period is, and how the certificate can be used.
public key infrastructure	Term that includes all the CSPs, Certificates, and standards for encryption and digital signatures using public/private key pairs. Also, an arrangement that provides for trusted third party vetting of, and vouching for, user identities. It also allows binding of public keys to users. This is usually carried out by software at a central location together with other coordinated software at distributed locations. The public keys are typically in certificates.
QC	See qualified electronic certificate.
QEC	See qualified electronic certificate.
QES	See qualified electronic signatures.
QSCP	See qualified certificate service provider.
qualified certificate service provider	A CSP that has met the high assurance requirements spelled out in the EU Signature Directive as well as the implementing Member State's legislation, and provides credentials under a high assurance mechanism that includes secure signature creation devices.
qualified electronic certificate	A digital certificate from a QCSP that conforms to the RFC 3739 specification. It contains a qc statement that simply states that it is a qualified certificate. These types of certificates meet the requirements of the EU Signature Directive.

Table 9 Security Terms

qualified electronic signatures	Electronic signatures made by an secure signature-creation device with a QEC provided by a QCSP.
roaming ID	A roaming ID is a digital ID that is stored on a server. The private key always remains on the server, but the certificate and its public key can be downloaded at the subscriber's request to any location. Roaming IDs require an Internet connection and require the user to authenticate to the server to initiate the signature process. They eliminate the need to provide hardware tokens to users for private key storage.
root certificate	The top-most issuing certificate in a certificate chain; sometimes used as a trust anchor.
RSA	An encryption algorithm used to create a digital signature. The acronym derives from the creator's last names. In this case Rivest, Shamir and Adelman.
Secure Identity Across Borders Linked (STORK)	An EU project to better leverage cross-border trust and usage of eIDs.
secure signature creation device	A high assurance hardware device (smart card, USB token, etc) that stores the private key associated often with a QEC.
security restricted property or method	A property or method whose availability is restricted to certain events such as batch processing, console execution, or application startup. For example, in Acrobat 7.0, a security-restricted method (S) can only be executed through a menu event if one of the following is true: The JavaScript user preferences item "Enable menu items JavaScript execution privileges" is checked or the method is executed through a trusted function. The *JavaScript for Acrobat API Reference* identifies the items that have restrictions.
signature algorithm	The combined usage of a digest method (e.g. MD5, SHA1) and an encryption algorithm (e.g. DSA or RSA)
signature image	A type of electronic signature where the signer signs a document by physically applying their handwritten signature to a document using a dedicated signature pad or Tablet PC and plugin to Acrobat or Reader. These signatures may also digitally sign the document at the time of signing to protect integrity.
SSCD	See secure signature creation device.
STF 364	Special Task Force #364 Group in ETSI/ESI working on PDF AdES or PAdES.
STORK	Secure Identity Across Borders Linked: An EU project to better leverage cross-border trust and usage of eIDs.
SubCA	See subordinate certificate authority.
subordinate certificate authority	A type of CA characterized by the fact that the ICA's certificate may itself be signed by a different ICA (thus 'subordinate' to it), all the way up to a root certificate.
timestamp	A digitally signed timestamp whose signer vouches for the existence of the signed document or content at the time given as part of the digital signature. The time stamp data can be embedded in the digital signature using a trusted time server (instead of the time clock of the computer that is used to apply the digital signature). See also TSP.
TL	See trust list.
Trust (Trust service) Status List	As described in ETSI TS 102 231, a format and method for communicating in human and machine readable form all of the certificates trusted by each member state and their QCSPs.
trust anchor	A certificate in a certificate chain that is trusted for selected operations. It could be an intermediate certificate authority rather than a root; that is, it does not have to be the topmost certificate in the chain. Certificates that chain up to this certificate will also be trusted for the same operations. It is usually issued by a 3rd party certificate authority.
trust list	A list of trusted CA's and certificates.

Table 9 Security Terms

TS	See timestamp.
TSL	See Trust (Trust service) Status List.
TSP	A timestamp protocol which is described in RFC 3161.
X.509v3	In cryptography, X.509 is an ITU-T standard for a public key infrastructure (PKI) for single sign-on (SSO) and Privilege Management Infrastructure (PMI). X.509 specifies, amongst other things, standard formats for public key certificates, certificate revocation lists, attribute certificates, and a certification path validation algorithm.
XAdES	See XML Advanced Electronic Signatures.
XML Advanced Electronic Signatures	An EU Advanced Electronic Signature Format relying on XML signatures, as described in ETSI TS 101 903.

Index

D

E

F

G

H

S

T

U

V

W

X