

Enhanced Security in Adobe Acrobat 9 and Adobe Reader 9

Updated for Adobe Acrobat and Reader 9.1

CONTENTS

Introduction	1
What is Cross-Domain Security?	2
Cross-Domain Access	3
User Preferences for Privileged Locations	8
FDF File Handling	10
Reader Configuration	11
Best Practices and Sample Use Cases	12
Resources	14

Introduction

PDF was originally a format used for final form representation for viewing, printing, and document interchange. With the addition of interactive form features, multimedia, and scripting, PDFs continue to become more capable with each release. To proactively address potential security issues, Adobe® Acrobat® 9 and Adobe Reader® 9 feature an Enhanced Security mode designed to provide advanced control over cross-domain data access and how FDF files are handled. Because the Enhanced Security restrictions may affect some legacy PDF workflows, Reader 9 also provides ways to increase privileges for documents and servers that can be fully trusted.

This document describes how Acrobat and Reader 9, with Enhanced Security turned on, prevents cross-domain data downloading, unless it is explicitly permitted by a *crossdomain.xml* file on the server, or by the user's setting of *privileged locations*. It also explains the new and more secure procedures for the handling of FDF files.

In addition, this document includes some sample use cases to describe how the security enhancements might affect some PDF workflows, and presents best practices for how to minimize the impact on those workflows.

NOTE: This document mainly refers to Adobe Reader, but all such statements apply equally to Adobe Acrobat.

Intended Audience

This document is intended for:

- system architects, IT managers, and workflow software developers who need an understanding of how the new security features might impact their software or company workflows.
- decision makers evaluating Adobe Acrobat, Reader, or LiveCycle for use in their workflows, who need to know technical details about security issues.
- developers creating interactive PDF content or forms based on PDF or LiveCycle forms technology.

Detailed instructions for using security features are contained in the Acrobat 9 and Reader 9 user and administration security guides (see [“Resources” on page 14](#)). All changes since the last version of this document are indicated with changebars appearing in the left margin.

What is Cross-Domain Security?

Over the years, Adobe Reader has added many new features, making it an extremely versatile tool capable of viewing, searching, digitally signing, verifying, printing, and collaborating, when using PDF documents. Creators of PDF can also include JavaScript to enable richer experiences to end-users. However, new capabilities sometimes come with new vulnerabilities, so Adobe proactively introduces new security functionality to combat emerging threats with each revision of the software.

One such area being introduced with Adobe Reader 9 is cross-domain security, which addresses a wide variety of potential threats to users and computer systems. However, before describing the new security features, it is important to discuss some background to the problem. That is followed by a short section on how Adobe products such as Flash have successfully built a strong security model, which is the basis for the new security features offered in Reader 9.

Web Security and Vulnerabilities

The ability to run a scripting language such as JavaScript or VBScript allows Web page authors to add a significant amount of features and interactivity to a Web page. However, this same capability can be abused by attackers, and the default configuration for most Web browsers enables scripting, which can introduce multiple vulnerabilities.

One of the earliest attempts to combat attacks was Netscape's *same-origin policy* introduced in the Netscape Navigator 2.0 Web browser. The policy prevents a document or script, loaded from one site of origin, from accessing resources loaded from another site of origin (see examples in [Table 1, "Same-Origin and Cross-Domain Examples,"](#) on page 5).

To counter the same-origin policy, a wide variety of attack patterns have evolved such as the Cross-Site Scripting (XSS), Cross-Site Request Forgery (XSRF), and many related variants. All of these attack patterns have one thing in common—they exploit the trust shared between a user and a website by circumventing the main protection mechanism: the same-origin policy.

While the same-origin policy has helped thwart many attack scenarios, it has fallen short in two different and distinct ways, which are often at odds with each other:

- Attack patterns such as XSS and XSRF can still be executed to exploit systems using scripting that communicates with other sites, and . . .
- Rich Internet Applications (RIAs) require the ability for the client to talk to multiple servers (domains) for improved user experiences.

Rich Internet Applications (RIAs), Adobe Flash, and Adobe Reader

Security is often at odds with usability. In fact, developers of Ajax applications are often faced with the same-origin policy limiting their functionality. To make matters worse, browsers all implement their same-origin policy slightly differently, which can lead to complex code, never bodes well for security, and usually results in a less-than-desirable user experience.

Adobe Flash began addressing this problem several releases ago by implementing and standardizing on a cross-domain security model that has evolved over the years into a robust, secure solution.¹ By providing controls for who may receive data from whom, Adobe Flash can power rich Internet applications that are safe and extremely flexible.

As Adobe Reader became more powerful over the years (i.e. support for JavaScript and Web Service interaction), the line between *document* and *application* gradually became more blurred, and Adobe began to leverage the Flash security model where appropriate. One such area is in Adobe Reader 9, with cross-domain security functionality. This is apparent in two main usage patterns with Adobe Reader:

- Reader in the Browser – When Adobe Reader is run from inside the browser, it uses the same operational behavior as Flash in the browser.
- Stand-Alone Reader – When Adobe Reader is run stand-alone, such as viewing a document on a local file system or opened as an attachment in e-mail. See the “[Stand-alone Reader](#)” section, below.

Cross-Domain Access

Reader 9, with Enhanced Security turned on, allows only same-origin data downloads to a PDF document. However, there are situations where it is desirable to allow cross-domain data transfer without such restrictions. The following two methods make it possible to bypass those restrictions:

- *using a crossdomain.xml policy file* — server administrators can post a file named `crossdomain.xml` which specifies which hosts can download data from the server (see below).
- *specifying privileged locations* — users can specify privileged locations (see “[User Preferences for Privileged Locations](#)” on page 8)

A `crossdomain.xml` file is an XML file which can specify either the URL for specific documents to which data can be sent, or a wildcard character (*) can be used to allow downloading to any remote file. The wildcard option is important for cases where the PDF is opened directly from a local disk by Reader 9, or a file referenced by a `file:URL` is opened from a local disk inside a browser. If the URL does not begin with `http:` or `https:`, and is referenced, for example, using one of the following:

```
file://  
\\server\folder\file.pdf  
C:\folder\file.pdf
```

(or related constructs), it has no domain.

Stand-alone Reader

The above discussion of cross-domain policies does not apply if a PDF is opened in a stand-alone Reader. Any local file that requests data from an `http[s]` server is

1. See <http://www.adobe.com/devnet/flash/security.html>

blocked unless that server has a cross-domain policy file containing a wild card or the local file is in a privileged location. This is consistent with Reader's cross-domain policy because the local filesystem (or Web-served e-mail message) does not have a domain, so access is only by permission stated in a crossdomain.xml file (also, see "User Preferences for Privileged Locations" on page 8).

Crossdomain.xml

crossdomain.xml is a file that is placed on a host which can be contacted by other hosts. It specifically lists the hosts that are allowed to make calls to its services (more precisely, it allows documents served from those hosts to make calls to its services). Crossdomain.xml even provides "*" functionality, which means any other host (including the "stand-alone" case) can make calls to its services. We should also note that the cross domain file itself is not just placed arbitrarily on a server, but placed at a particular base URL location on the server under which it will have the desired effect for the services available. For example, Adobe Flash has implemented this model and many leading sites are already using crossdomain.xml.

Taking a look at the crossdomain.xml file used on www.youtube.com:

```
<!-- http://www.youtube.com/crossdomain.xml -->
- <cross-domain-policy>
<allow-access-from domain="*.youtube.com"/>
<allow-access-from domain="*.ytimg.com"/>
<allow-access-from domain="*.google.com"/>
</cross-domain-policy>
```

it can be seen that any document hosted by a server on *.youtube.com, *.ytimg.com or *.google.com, can receive data from www.youtube.com. Specifically, Reader is asking the question: has the server that is providing the data agreed that the document is allowed to have it?

Table 1, "Same-Origin and Cross-Domain Examples" shows examples of combinations of PDF and server locations for both same-origin and cross-domain data requests (all examples are for PDF being viewed in a browser):

Same-origin: Reader allows data access (examples with a "No" in the third column below)

Cross-domain: crossdomain.xml file required. (examples with a "Yes" in third column below).

TABLE 1 Same-Origin and Cross-Domain Examples

PDF Location:	Received data location:	Crossdomain.xml required?	Example
c:\test.pdf	c:xyz.data	No	N/A
c:\test.pdf	smb://smb_server/xyz.data	No	N/A
\\smb_server\test.pdf	smb://smb_server/xyz.data	No	N/A
\\smb_server\test.pdf	c:\xyz.data	No	N/A
c:\test.pdf	http://xyz.com/xyz.data	Yes	<allow-access-from domain="*" />
\\smb_server\test.pdf	http://xyz.com/xyz.data	Yes	<allow-access-from domain="*" />
http://www.xyz.com/test.pdf	http://www.xyz.com/xyz.data	No	N/A
http://www.xyz.com/test.pdf	https://www.xyz.com/xyz.data	Yes	See Note 1 (below)
https://www.xyz.com/test.pdf	http://www.xyz.com/xyz.data	Yes	See Note 2 (below)
http://www.xyz.com/test.pdf	http://www.xyz.com:8080/xyz.data	Yes	See Note 1 (below)
http://www.xyz.com/test.pdf	http://www.xyz.com/xyz.data	No	N/A
http://www.xyz.com/test.pdf	http://xyz.com/xyz.data	No	N/A
http://www.xyz.com/test.pdf	http://bar.com/xyz.data	Yes	See Note 1 (below)
http://www.xyz.com/test.pdf	http://www.bar.com/foo.data	Yes	See Note 1 (below)

Note 1: The following examples show typical entries that might be used in a crossdomain.xml file, as shown in Table 1 above:

```
<allow-access-from domain="www.xyz.com" />
<allow-access-from domain="*.xyz.com" />
<allow-access-from domain="*" />
```

Note 2: This case of a secure host receiving data from a non-secure host requires the same statements shown in Note 1 plus the following:

```
<allow-access-from domain="*.xyz.com" secure="false" />
```

If it is not desirable to place the crossdomain.xml file at the root of the Web server, it may be placed at an arbitrary location if it is specified by using a call in the document to the JavaScript method: `app.loadPolicyFile(url);`

It is important to note that cross-domain restrictions apply only to the downloading of data, not to uploading, where downloading is defined as a document receiving data from a server. Also, cross-domain restrictions only apply when the data is being requested by, and is subsequently accessible to, a document, and not just requested by Reader. For example, if Reader requests a timestamp from a remote server to validate a digital signature, it is not restricted because the data was not requested by the document.

A cross-domain check is required for at least the following cases:

- Form submissions (if data is returned)
- SOAP requests
- References to streaming media (audio, video)
- `Net.HTTP.request()`

The following are cases where checking `crossdomain.xml` is not required:

- A form sends data to a server, but no data is sent back into the form.
- A form sends data to a server on the same host as the one on which the form is hosted, and receives data back from that server.
- A user clicks on a hyperlink; which is considered sending data, not downloading (although the user may get a security warning dialog).
- A user participates in a shared workflow such as a document review using a collaboration server. The dialog presented to the user is considered adequate warning that downloading may occur.
- Enhanced Security is turned off.
- Enhanced Security is turned on, but document is in a privileged location.\

Support for Local Files

Enhanced security in version 9.0 depended on establishing trust between a particular server and the author/origin of the document. Usually this is handled by the server hosting a cross-domain policy file that lists the domains it will allow. However, with local files, that approach will not work because local files are considered to not have a domain.

Acrobat and Reader 9.1 introduce an extension to the cross-domain policy to uniquely identify a trust relationship between a server and a document stored on a user's local file system. The trust is created by inserting the form's *fingerprint* — the SHA-1 hash value of its certification signature, into the `crossdomain.xml` policy file for the server that the document will try to access. That can only be done for forms that are certified or Reader-enabled.

To extract the fingerprint from a certified or Reader-enabled form and apply it to the server's `crossdomain.xml` policy file:

1. For **Adobe Reader**:

- 1a. Select Document > Manage Trusted Identities
- 1b. Select Certificates from the Display drop-down menu
- 1c. Select the certificate for the desired document
- 1d. Click the Show Certificate button to bring up the Certificate Viewer window

Continue to Step 2.

For **Adobe Acrobat**:

- 1a. Select Advanced > Security Settings
 - 1b. Select the Certificate Details icon at the top menu bar to bring up the Certificate Viewer dialog window.
2. Select the Details tab in the Certificate Viewer to see the list of all data for the selected certificate.
3. In the Certificate Data pane, you will see the fingerprint portion which has the name "SHA1 digest". When this item is selected, the data pane below it will display the 40-byte fingerprint.

4. Select the hex data fingerprint and copy it to the clipboard.
5. Paste the data into the `crossdomain.xml` file using the guidelines given in [“Crossdomain.xml File Syntax for Local Files” on page 7](#).

Note that signatures optionally allow some changes to a document without invalidating the signature:

- Rights-enabled PDF documents may allow form fill-in, digital signatures, commenting and/or file attachments.
- Certified PDF documents may allow form fill-in, digital signatures, annotations, form fill-in and digital signatures.

If a document is changed, but the changes are allowed, the signature is valid for cross-domain checking purposes.

The following are examples where Acrobat or Reader does not need to do a cross-domain check for a local PDF file:

- Trusted folders/Web sites take precedence over cross domain checks. If the file is already trusted, no cross domain checks will be performed.
- If a document is certified and the certificate is trusted to do “Privileged System Operations”, no cross domain checks will be performed.
- If a user certifies a document with his/her own certificate, the certificate is always trusted to do “Privileged System Operations” so no crossdomain checks are performed. (These rights are only accorded certifying, not regular, signatures.)

Crossdomain.xml File Syntax for Local Files

The following syntax is used for specifying signatures in policy files:

```
<allow-access-from-identity>
  <signatory>
    <certificate
      fingerprint="01:23:45:67:89:ab:cd:ef:01:23:45:67:89:ab:cd
        :ef:01:23:45:67"
      fingerprint-algorithm="sha-1"/>
    </signatory>
  </allow-access-from-identity>
```

There must be exactly one `<signatory>` block inside an `<allow-access-from-identity>`, and there must be exactly one

`<public-key>` or `<certificate>` block inside a `<signatory>`.

To permit multiple signatories, a policy file author simply writes multiple `<allow-access-from-identity>` blocks.

Though there are two types of legal entries inside the `<signatory>` block, `<certificate>` and `<public-key>`, Acrobat and Reader 9.1 will support the `<certificate>` entry only, as follows:

- Case is ignored in the fingerprint string. The blank and colon characters are ignored within the fingerprint string. The `<fingerprint-algorithm>` entry is required. That is, if the `<fingerprint-algorithm>` entry is missing or malformed, this certificate block will be skipped. For 9.1, the only supported

value for `<fingerprint-algorithm>` is "SHA-1"; the comparison is casesensitive.

- SHA-1 hashes must be exactly 40 chars long (not counting spaces or colons).
- Since `<public-key>` is not supported for 9.1, any `<public-key>` blocks encountered will be skipped. (This means there is no parsing of the `<public-key>` block.)

Sample Workflow for Creating Trust for a Distributed Form:

1. Form is a certified or Reader-enabled PDF (which is not in a trusted location and the certificate is not trusted).
2. Administrator extracts the fingerprint (e.g. SHA1 hash of the entire certificate) from the form's certificate.
3. The fingerprint data is added to the `crossdomain.xml` file on the server.
4. User opens the form and performs an action that requires requesting data from a server, which is allowed because Acrobat/Reader confirms that the server's `crossdomain.xml` file contains the fingerprint of that document.
5. Operation succeeds.

Web Service Proxy

When forms are opened in Reader 9 in the browser, Web service calls back to the origin serving the document is allowed by default in the cross-domain security policy and therefore does not require configuration of a `crossdomain.xml` at the site of the Web service. This is already a very typical pattern in LiveCycle deployments. Therefore, if an organization is thinking about calling Web services on other hosts, they could instead employ the Web Service Proxy pattern. In this design pattern, new Web services are authored using LiveCycle at the same origin as the hosted document, which then acts as a proxy to call the third-party Web service (similar to Web mash-ups).

Data Taint

The cross-domain security policy also provides a better method for dealing with *data taint* restrictions (supported in several previous versions of Reader), which prevents data from being acquired from multiple sources and sent elsewhere.

Taint and cross-domain policies deal with the same problem, but cross-domain does so in a superior way because it places the policy decisions on the owner of the data, not the user of the client application as the taint mechanism does.

If cross-domain support is enabled in Reader 9 (that is, Enhanced Security is *On*), the taint restriction is turned off and cross-domain is used instead.

User Preferences for Privileged Locations

Reader 9 provides a method for a user or a system administrator to increase the privileges of chosen files, folders, and hosts by specifying them as *privileged*

locations. Files in these privileged locations are exempted from certain security policies and therefore, for example, can be exempt from cross-domain security. The locations can be set by the user through the UI (Preferences dialog), or by a system administrator using Adobe Customization Wizard to specify installer settings prior to enterprise-wide deployment.

The three choices for privileged locations are:

Files — A file is defined by a path, so its security settings will be invalid if that file is moved. The difference between privileged PDF files and folders is the number of files. If a user has a large number of files they know they can trust, it may be more practical to put them all in one privileged PDF folder. Conversely, a user may use privileged PDF files if they have many PDFs in a single folder but only want to trust two of them.

Folders — Privileged PDF folders are similar to privileged PDF files except that all files in a specified folder (but *not* in sub-folders) have the same privileges.

Host — A privileged PDF site is appropriate for PDFs that can be opened in a browser from a Web server. A privileged PDF host can only be specified at the host name level; for example, `www.adobe.com` can be specified, but not `www.adobe.com/products/`. The specified host must be complete with no wild cards (unlike for `crossdomain.xml` files). The user will have the option to only specify that the host connection must be secure, for example, that it must be an `https:` connection. All PDFs on the specified host will all have the same privileged PDF settings.

Using privileged locations, the user can bypass the security restrictions on the following, which would otherwise be in effect:

- Cross-domain data access
- Silent printing
- External streams access
- Document JavaScript sending data to a remote server
- FDF data injection
- FDF script injection
- Data taint: when data is downloaded from multiple hosts and then sent to another host

If a file, folder, or host is specified as being a privileged location, all of the above restrictions are bypassed.

Admin and User Lists of Privileged Locations

Reader 9 maintains two lists of privileged PDF files and locations: *Admin List* and *User List*. The Admin List is shared by all users on the system, and it should be set up by a system administrator (manually or by using Adobe Acrobat Customization Wizard or other administrative tools, before deployment/installation). The User List is for the current user only, and it can be edited for the current user from the Reader UI. The default installation does not specify any privileged locations.

FDF File Handling

Forms Data Format (FDF) files are generally used for transmitting and storing forms data. The most common uses of an FDF file, as relevant to Reader 9 enhanced security, are:

- To open a specified PDF
- To inject data into the specified PDF (for example, form field data)
- To inject and/or run a script within the context of a specified PDF

There are other tasks for which FDFs are used, for but they are not affected by the restrictions introduced in Reader 9.

The restrictions for the following operations: opening an FDF and injecting data or scripts, pertain only to what are called *Initiating FDF* files—ones that initiate a set or sequence of events, such as for form submission or actions for collaborative reviews. They are generally unsolicited, but not necessarily unexpected, by the user.

Open a Specified File

FDF files can have a `/F` (or `/UF`) key that specifies a target PDF to be opened and potentially acted upon.

Policies (strictly for the 'Open target PDF' phase):

- If the FDF is a local file (that is, not using `http:` or `https:`) and the target PDF is also a local file, opening that PDF is not blocked and no user authorization is required.
- If the FDF is a local file and the target PDF is hosted on an `http://` or `https://` server, opening that PDF requires a security warning dialog.
- If the FDF is hosted on an `http:` or `https:` server, and the target PDF is also on an `http:` or `https:` server, opening that PDF is not blocked and no user authorization is required.
- If the FDF is hosted on an `http:` or `https:` server and the target PDF is a local file, opening the PDF is blocked.

NOTE: XFDF and XDP files use the same rules as FDF with the following exceptions: XFDF does not support script injection; and XDP is only affected by these rules if the PDF is externally referenced (not embedded).

Data Injection

FDF files may be used to set the values of specified key/value pairs or to inject data form attributes into the target PDF.

The following examples are scenarios where FDF data injection will initiate a user-authorization dialog (there is no 'remember this' feature for these dialogs, so each data injection will force its own dialog):

- The user submits from a PDF in the browser, and the URL has `#FDF` at the end, and the FDF that comes back has an `/F` key pointing to a different PDF, which

needs to be loaded (all happening in the browser), and the FDF data gets injected into the second PDF.

- Same as above, except it all happens in stand-alone mode (that is, not in the browser); in that case the #FDF at the end of the URL is not needed.
- The “Initiating FDF” case. In the browser, an FDF comes in unsolicited (meaning you were in an HTML page before, and Acrobat may not be open yet) and initiates an action, and the FDF has a /F key for a PDF that it needs to open and populate with data. Also, Initiating FDFs can be local (it is not always browser-based). For example, if you get an FDF in an e-mail and double-click it.

Script Injection

FDF files can contain scripts that are injected into a PDF and run in the context of that PDF. For example, a validation script might be run to check for a valid date in a specific field, or it might be a “run after” script that executes as soon as the PDF is opened and injects comments into the PDF, or contacts a collaboration server.

In Reader 9.0, that is not allowed and FDF script injection will be blocked. To bypass this block, the user can specify a privileged location for that FDF file (see “User Preferences for Privileged Locations,” above). A privileged FDF will be allowed to inject the script in the context of the targeted PDF (assuming it is allowed to open it based on the “Open a Specified File” policies above). The target PDF in which the script will be run does not need to be privileged because it did not request cross-domain data. It does not inherit the privileged status; the reason is just that it did not initiate the operation, and the user has placed full trust in the FDF by specifying it as a privileged location.

Reader Configuration

Organizations can configure Adobe Reader before re-distributing it to their end-users. In fact, Adobe Reader registry settings are available for controlling many aspects of the product, so organizations can even push out configuration changes to an installed base as well. Two important ways to control cross-domain security is by configuring Reader’s privileged locations or certified documents.

- Privileged Locations: PDF or FDF files in privileged locations are exempted from cross-domain security checks.
- Organizations can also leverage a trust model applied to certified PDF documents. In this way, if a document is certified by a trusted certificate, Reader can be configured so the document can will be exempt from cross-domain security.

The configuration can be done using the Adobe Customization Wizard, which can be used to customize the installation and to specify the desired application and registry settings, including whether the user is allowed to change their settings or not. For more information, see:

http://www.adobe.com/go/enterprise_deployment

If the organization does not have control over the desktop, then manual configuration could be employed. This can be managed with smaller user communities, but is difficult in larger populations.

The UI in Reader 9 gives a simple view into a user's security configuration as based on settings done through the UI. More detailed security settings can be specified using Adobe Customization Wizard, which can affect many more registry settings than is possible through the UI.

Best Practices and Sample Use Cases

Understanding Reader 9 behavior with Enhanced Security turned on and the optional ways of overriding this behavior is best described with several use cases, which we hope will describe current best practices in this area.

Internal Workflows

Internal workflows are defined as processes in which all users' desktops are under the organization's control. Examples of internal workflows are employee reviews, expense reports and purchase order requisitions.

Reader in the Browser: Best Practices

As a best practice, it is advisable to centralize all Web services at a common origin. Then, configure an explicit `crossdomain.xml` at the origin to reference the servers hosting your documents. If you cannot centralize your Web services at a common origin, then try to employ the use of the "path to origin" pattern or minimize the number of domains as much as possible. Otherwise, each distinct server hosting your Web services will need a similar `crossdomain.xml`.

Stand-Alone Reader: Best Practices

In the case of stand-alone Reader, there is no domain associated with the opened document. Therefore, if Web service calls are to be made from that document, the Web service host will need a `crossdomain.xml` file containing "*" as a way of trusting all documents. Adobe proposes using a `crossdomain.xml` with a "*" *only if*:

1. The Web service does not return sensitive or private data or,
2. The Web service is already using a form of user interactive authentication to govern requests for data.

But, if the Web service is not protected by authentication *and* it returns sensitive data then we strongly recommend that you secure your Web Service properly.

Another approach for the stand-alone case is to use privileged locations or certified documents to grant specific files the ability to make the Web service call. The result is that the document's privileges are escalated, and therefore should only be done for documents that the organization has control over.

Use Case #1: Travel Approval Form

An example of a form that might make benign Web service calls (that is, sensitive/private data is not returned), is a travel approval form that uses zip code / state auto-complete functionality. In both the browser and stand-alone case, a "*" placed in the crossdomain.xml file will work appropriately.

Use Case #2: Employee Salary Review

In this case, a Web service is used to pull current salary information from an SAP server system. Since the Web service requires authentication, again, it would be acceptable to place a "*" in the crossdomain.xml file.

Notice, in both of these use cases, stronger controls could be enabled by making the crossdomain.xml file more specific and not use the "*". In this case, workflows that are usually performed from within the browser will work as always, but the stand-alone case will not allow those calls unless further methods are implemented (privileged locations or certified documents) as stated before.

External Workflows

External workflows are defined as processes in which some users' desktops are *not* under the organization's control. Examples of external workflows are tax forms, loan documents and complaint forms.

Reader in the Browser: Best Practices

As a best practice, Adobe currently recommends the usage of Reader in the browser for external workflows.

Since it can be extremely difficult to consolidate all Web services at a single origin with external workflows, making sure all Web service domains have properly configured crossdomain.xml files can be challenging. Therefore, Adobe recommends consolidation as much as possible and employing the "path to origin" pattern when possible.

If you cannot use the "path to origin" pattern, then the next step is to see if a "*" can be placed on the Web service that is outside the pattern. Again, using a "*" is only acceptable if either of the following is true:

- The Web service does not return sensitive or private data, or,
- The Web service is already using a form of authentication to filter incoming requests

Use Case #3: Traffic Court Form

An example of a form that might make benign Web service calls (that is, sensitive or private data is not returned), is a traffic court form. Like in Use Case #1, such a form might use zip code / state auto-complete functionality. In both the browser and stand-alone case, a "*" placed in the crossdomain.xml file will allow the Web service call to be completed. The "*" will overcome the any extra requirements of enabling trust of the stand-alone case.

Use Case #4 Streaming Media

Acrobat 9 allows customers to include any content type that can be natively rendered by Flash, since the Flash runtime is now part of Acrobat. This includes video such as .flv, H.264, and audio: MP3 and AAC. Since these formats can include files that are quite large, they may be referenced by their URL in the PDF.

An example would be a content developer who creates promotional PDF brochures for a major publisher. The PDFs promote various books that have been published and each page includes links to MP3 files that are short little podcasts from the various authors. These PDFs are downloaded to a user's local disk and opened outside the browser, so the PDF is not on the same domain as the MP3.

If you click on one of these links in Acrobat 9 with Enhanced Security enabled, you get a dialog indicating that the publisher's server hosting the MP3 domain does not have the proper crossdomain.xml file and you cannot listen to the MP3. The workaround is to add the PDF to your list of trusted files, but another solution would be for the publisher to simply add a crossdomain.xml file with "*" on their server.

Resources

(The following links currently point to Acrobat and Reader 8 documents and are placeholders for documents to be released for version 9)

JavaScript

JavaScript for Acrobat API Reference:

http://www.adobe.com/go/acrobatsdk_js_developers

Developing Acrobat Applications with JavaScript:

http://www.adobe.com/go/acrobatsdk_js_apis

Security

The following documents on PDF security and digital signatures are all available at:

http://www.adobe.com/go/acrobat_security

User and Administrator Guides

Security Administration Guide for Adobe Acrobat and Adobe Reader Version 9

Digital Signature User Guide for Adobe Acrobat and Adobe Reader Version 9

Document Security User Guide for Adobe Acrobat and Adobe Reader Version 9

Copyright 2008 Adobe Systems, Incorporated. All rights reserved.

Adobe Systems Incorporated

345 Park Avenue, San Jose, CA 95110-2704 USA

<http://www.adobe.com>

Adobe, the Adobe logo, Acrobat, Adobe LiveCycle, and Reader are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States and/or other countries. Mac OS is a trademark of Apple Computer, Inc., registered in the United States and other countries. Linux is a registered trademark of Linus Torvalds. Microsoft, Windows, Windows Vista, and Word are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries. All other trademarks are the property of their respective owners.

15 December 2008

