



WHITE PAPER

Macromedia Flash Player 8 Security-Related APIs

by Adrian Ludwig

September 2005

Copyright © 2005 Macromedia, Inc. All rights reserved.

The information contained in this document represents the current view of Macromedia on the issue discussed as of the date of publication. Because Macromedia must respond to changing market conditions, it should not be interpreted to be a commitment on the part of Macromedia, and Macromedia cannot guarantee the accuracy of any information presented after the date of publication.

This white paper is for information purposes only. **MACROMEDIA MAKES NO WARRANTIES, EXPRESS OR IMPLIED, IN THIS DOCUMENT.**

Macromedia may have patents, patent applications, trademark, copyright or other intellectual property rights covering the subject matter of this document. Except as expressly provided in any written license agreement from Macromedia, the furnishing of this document does not give you any license to these patents, trademarks, copyrights or other intellectual property.

Macromedia, Flash, Breeze, Central, ColdFusion, FlashCast, Flash Lite, Flex, and MXML are either trademarks or registered trademarks of Macromedia, Inc. in the United States and/or other countries. The names of actual companies and products mentioned herein may be the trademarks of their respective owners.

Macromedia, Inc.
601 Townsend Street
San Francisco, CA 94103
415-832-2000

Contents

- Introduction1**
- Loading operations2**
- Security-specific APIs3**
- SWF file loading.....4**
- Data loading6**
- Data sending9**
- Shared object references 11**
- Inter-SWF file (or cross-scripting) APIs..... 12**
- Outbound scripting 13**
- Other sensitive APIs 14**
 - User Agent string..... 14
 - Systems capabilities APIs 14
 - User settings APIs.....17
 - Camera17
 - Audio In (Microphone)..... 18
 - Clipboard..... 18
 - Mouse and keyboard APIs 19
- Flash Player updates 20**

Introduction

Macromedia Flash Player runs Flash applications (also referred to as SWF files). Flash content is delivered as a series of instructions in binary format to Flash Player over web protocols in the precisely described SWF (.swf) file format. The SWF files themselves are typically hosted on a server and then downloaded to, and displayed on, the client computer when requested. Most of the content consists of binary ActionScript instructions. ActionScript is the ECMA standards-based scripting language used by Flash that features APIs designed to allow the creation and manipulation of client-side user interface elements and for working with data.

This document lists all APIs related to Flash Player (both ActionScript APIs and others) that are affected by the Flash Player security architecture.

For a detailed description of the Flash Player security architecture, see the *Macromedia Flash Player Security* document.

This document is intended for developers (including programmers and other authors) designing and publishing Flash applications, who are concerned with the security aspects of the Flash applications they develop.

This document assumes that the reader is familiar with Flash and ActionScript, and their related terms, authoring tools, and environments.

Readers must be familiar with the following concepts, each of which is described in the *Macromedia Flash Player Security* document:

- *Security sandboxes* that Flash Player uses to classify the source domain of a SWF file or other resource
- The Flash Security Manager and the Flash Settings dialog boxes (used by a Flash user to set security-related settings)
- The `mms.cfg` file (used by administrators and users to make Flash security-related settings for a specific computer or user on that computer)
- The Global Trust and User Trust directories (used by administrators and users to make Flash security-related settings for a specific computer or user on that computer)
- Cross-domain policy files used by website administrators to control Flash application access to resources on a server

Readers should also have a good knowledge of Flash content development and ActionScript.

Many of the APIs listed in this document describe security checks, and permission mechanisms. A *security check* is a process performed by Flash Player that cannot be overridden by any stakeholder (including the end user, administrators of the client computer, website administrators, or the Flash developer). A *permission mechanism* is a security-related process that can be modified by a specific stakeholder (such as an administrator, an end user, a website administrator, or the Flash developer). Changes in the permissions mechanisms may influence the success or failure of a call to the API.

Loading operations

Flash Player identifies all APIs in the ActionScript standard library that fall into any of the security-related categories (detailed further in the following sections), and makes the appropriate security decisions whenever any of those APIs are called. However, there are some differences due to the finer granularity of the sandboxes in Flash Player 8, some new features in Flash Player 8, and other enhancements.

The following table summarizes the various SWF file-loading operations available for ActionScript. Although the intended purposes and effects of each operation could involve one or more of the columns, not all the columns apply in each case (N/A = Not Applicable).

Table 1: Loading operations summary

Operation	Play as media		Media library		Script library		Permissions needed
	Display objects created?	Sandbox of display objects	Media symbols made available?	Sandbox of symbol instances	Script made available?	Sandbox of script	
loadMovie()	Yes, if loadee has any on the Stage	Loadee	N/A	--	Yes	Loadee	Loading: none Calling script: allowDomain
RSL import into Flash	N/A	--	Yes: loadee must export and loader must import	Loadee	Yes: button and clip actions of instances	Loadee	Policy file, because loader could enslave loadee script

Security-specific APIs

The ActionScript `System.security` class contains the following methods and properties that let SWF files determine security-related information. The `System.exactSettings` property is another security-specific API. This table briefly summarizes the security-specific APIs; for more information, see the relevant entries in the *ActionScript 3.0 Language Reference*.

Table 2: Security-Specific APIs

API	Description
<code>System.security.sandboxType</code>	<p>Added in Flash Player 8. Indicates the type of security sandbox in which the calling SWF file is operating. <code>System.security.sandboxType</code> has one of the following values:</p> <ul style="list-style-type: none"> ▪ <code>remote</code>: This SWF file is from an Internet URL and will operate under domain-based sandbox rules. ▪ <code>localWithFile</code>: This SWF file is a local file, but it has not been trusted by the user and was not published with a networking designation. This SWF file may read from local data sources, but may not communicate with the Internet. ▪ <code>localWithNetwork</code>: This SWF file is a local file and has not been trusted by the user. It was published with a networking designation. This SWF may communicate with the Internet, but may not read from local data sources. ▪ <code>localTrusted</code>: This SWF file is a local file and has been trusted by the user, using either the Settings Manager or a <code>FlashPlayerTrust</code> configuration file. This SWF file may both read from local data sources and communicate with the Internet.
<code>System.security.allowDomain()</code>	<p>Lets SWF files and HTML files in the identified domains access objects and variables in the SWF file that contains the <code>allowDomain()</code> call.</p> <p>For more information, see the <code>System.security.allowDomain()</code> entry in the <i>ActionScript 3.0 Language Reference</i>.</p>
<code>System.security.allowInsecureDomain()</code>	<p>Lets SWF files and HTML files in the identified domains access objects and variables in the calling SWF file, which is hosted by means of the HTTPS protocol. Macromedia does not recommend using this method.</p> <p>For more information, see the <code>System.security.allowInsecureDomain()</code> entry in the <i>ActionScript 3.0 Language Reference</i>.</p>
<code>System.security.loadPolicyFile()</code>	<p>Loads a cross-domain policy file from a location specified by the <code>url</code> parameter. Flash Player uses policy files as a permission mechanism to permit Flash movies to load data from servers other than their own.</p> <p>For more information, see the <code>System.security.loadPolicyFile()</code> entry in the <i>ActionScript 3.0 Language Reference</i>.</p>

System.exactSettings	In Flash Player 6, player settings are chosen based on a SWF's superdomain; for example, SWF files from www.example.com and store.example.com would both use the player settings for example.com. In Flash Player 7 and later, security settings are chosen by default based on a SWF file's exact domain; for example, a SWF file from www.example.com would use the player settings for www.example.com, and a SWF file from store.example.com would use the separate player settings for store.example.com. When security.exactSettings is set to true, the Flash Player uses exact domains for player settings. When it is set to false, the Flash Player uses superdomains for player settings.
----------------------	--

SWF file loading

A SWF file that is running in Flash Player can load other SWF files as movie clips within the active file.

The following table describes the APIs that provide for loading SWF files, and includes information about relevant security checks, permission mechanisms, and error notifications.

Table 3: SWF file loading APIs

API	Description
MovieClip. loadMovie()	<p>Loads a SWF file (or graphic) into Flash Player.</p> <p><i>Security checks:</i></p> <ul style="list-style-type: none"> ▪ Loading is not allowed if the calling movie clip is in the local-with-file-system and the loaded movie clip is from the network or local-with-networking sandbox. ▪ Loading is not allowed if the calling SWF file is in a network sandbox and the loaded movie clip is local. <p><i>Permission mechanisms:</i></p> <ul style="list-style-type: none"> ▪ Policy File-network sandbox access from the local trusted or local-with-networking sandbox requires permission from website. ▪ Default for other cross-domain requests is to allow access. <p><i>Error notifications:</i></p> <ul style="list-style-type: none"> ▪ SecurityError exception

API	Description
<p>MovieClip. loadMovieNum()</p>	<p>Loads a SWF file (or graphic) into a specific level in Flash Player.</p> <p><i>Security checks:</i></p> <ul style="list-style-type: none"> ▪ Loading is not allowed if the calling SWF file is in the local-with-file-system sandbox and the loaded movie clip is from a network sandbox or the local-with-networking sandbox. ▪ Loading is not allowed if the calling SWF file is in a network sandbox and the loaded movie clip is local. <p><i>Permission mechanisms:</i></p> <ul style="list-style-type: none"> ▪ Policy File–network sandbox access from the local trusted or local-with-networking sandbox requires permission from the website. ▪ The default for other cross-domain requests is to allow access. <p><i>Error notification:</i></p> <ul style="list-style-type: none"> ▪ SecurityError exception
<p>MovieClipLoader. load()</p>	<p>Another method for loading a SWF file (or graphic) into Flash Player.</p> <p><i>Security checks:</i></p> <ul style="list-style-type: none"> ▪ Loading is not allowed if the calling SWF file is in the local-with-file-system sandbox and the loaded movie clip is from the network or local-with-networking sandbox. ▪ Loading is not allowed if the calling SWF file is in a network sandbox and the loaded movie clip is local. <p><i>Permission mechanisms:</i></p> <ul style="list-style-type: none"> ▪ Policy File; Cross-domain access from the local-trusted or local-with-networking sandbox requires permission from the website. ▪ The default for other cross-domain requests is to allow access. <p><i>Error notifications:</i></p> <ul style="list-style-type: none"> ▪ SecurityError exception

Data loading

Data loading is when ActionScript code loads data from sources other than Flash sources, such as XML documents or sockets. This is allowed when the Flash application making the request is in the same sandbox as the resource it is accessing. For other access attempts between different sandboxes, requests are forbidden by default, but permission can be granted by the administrator, user, or website that controls the requested resource.

The following table describes APIs that load data from files other than SWF files, and it includes information about relevant security checks, permission mechanisms, and error notifications:

Table 4: Data loading APIs

API Names	Description
<p>LoadVars.load() LoadVars.sendAndLoad() XML.load() XML.sendAndLoad()</p>	<p>Loads text or XML data from a server.</p> <p><i>Security checks:</i></p> <ul style="list-style-type: none"> ▪ Not allowed if the calling SWF file is in the local-with-file-support sandbox and resource is from a network or the local-with-networking sandbox <p><i>Permission mechanisms:</i></p> <ul style="list-style-type: none"> ▪ Policy File—The default is to deny access between sandboxes. The website can enable access to a resource by adding a policy file. <p><i>Error notification:</i></p> <ul style="list-style-type: none"> ▪ SecurityError exception or SecurityError event
<p>XMLSocket.connect()</p>	<p>Connects to a socket server for low-overhead, low-level data exchange.</p> <p><i>Security checks:</i></p> <ul style="list-style-type: none"> ▪ Not allowed if the calling SWF file is in the local-with-file-support sandbox. ▪ Not allowed if the port number is less than 1024. <p><i>Permission mechanisms:</i></p> <ul style="list-style-type: none"> ▪ Policy File—The default is to deny access between sandboxes. The website can enable access to a resource by adding a policy file with the XMLSocket protocol. The author can invoke loadPolicyFile(). <p><i>Error notification:</i></p> <ul style="list-style-type: none"> ▪ SecurityError exception or SecurityError event from XMLSocket object

API Names	Description
<p><code>NetConnection.connect()</code></p> <p>(AMF Remoting only)</p>	<p>Sets up an AMF (ActionScript Message Format) object for RPC (remote procedure call) calls.</p> <p><i>Security checks:</i></p> <ul style="list-style-type: none"> ▪ Not allowed if the calling SWF file is in the local-with-file-support sandbox. <p><i>Permission mechanisms:</i></p> <ul style="list-style-type: none"> ▪ Policy File–The default is to deny access between sandboxes. The website can enable access to a resource by adding a policy file. <p><i>Error notification:</i></p> <ul style="list-style-type: none"> ▪ <code>SecurityError</code> exception
<p><code>NetConnection.connect()</code></p> <p>(Flash Communication Server RTMP only)</p>	<p>Connects to Flash Communication Server for data exchange.</p> <p><i>Security checks:</i></p> <ul style="list-style-type: none"> ▪ Not allowed if the calling SWF file is in the local-with-file-support sandbox. <p><i>Permission mechanisms:</i></p> <ul style="list-style-type: none"> ▪ Server-side ActionScript–The default is to allow access. The website can deny access to a resource by adding server-side ActionScript application logic in Flash Communication Server. <p><i>Error notification:</i></p> <ul style="list-style-type: none"> ▪ <code>SecurityError</code> exception or server response
<p><code>NetStream.play()</code></p>	<p>When used with a null <code>NetConnection</code>, loads a Flash Video (FLV) file and plays it.</p> <p><i>Security checks:</i></p> <ul style="list-style-type: none"> ▪ Not allowed if the calling SWF file is in the local-with-file-support sandbox and the resource is in a non-local sandbox <p><i>Permission mechanisms:</i></p> <ul style="list-style-type: none"> ▪ Policy File–Network sandbox access from local trusted or local-with-networking sandbox requires permission from the website ▪ The default for other cross-domain requests is to allow access. <p><i>Error notification:</i></p> <ul style="list-style-type: none"> ▪ <code>SecurityError</code> exception

API Names	Description
<p>Sound.load()</p>	<p>Retrieves an MP3 file for playback.</p> <p><i>Security checks:</i></p> <ul style="list-style-type: none"> ▪ Not allowed if the calling SWF file is in the local-with-file-support sandbox and the resource is non-local <p><i>Permission mechanisms:</i></p> <ul style="list-style-type: none"> ▪ Policy File–Network sandbox access from local trusted or local-with-networking sandbox requires permission from website ▪ Default for other cross-domain requests is to allow access. <p><i>Error notification:</i></p> <ul style="list-style-type: none"> ▪ SecurityError exception
<p>FileReference.upload() FileReference.download()</p>	<p>Uploads (sends) or downloads (loads) files to or from a server. The upload() method is new in Flash Player 8.</p> <p><i>Security checks:</i></p> <ul style="list-style-type: none"> ▪ Not allowed if the calling SWF file is an untrusted local file. <p><i>Permission mechanisms:</i></p> <ul style="list-style-type: none"> ▪ Policy File–The default is to deny access between sandboxes. A website can enable access to a resource by adding a policy file. <p><i>Error notification:</i></p> <ul style="list-style-type: none"> ▪ SecurityError exception or SecurityError event from a FileReference object

Data sending

Data sending is when ActionScript code from a SWF file sends data to a server or resource. Data sending is generally allowed. (For more information, see the *Macromedia Flash Player Security* document.) The following table describes the APIs that support sending data to servers (where allowed), and includes information about relevant security checks, permission mechanisms, and error notifications:

Table 5: Data Sending APIs

API Name	Description
LoadVars.send() XML.send()	<p>Sends data to a server and opens a browser window to display the server response.</p> <p><i>Security checks:</i></p> <ul style="list-style-type: none">▪ Not allowed if the calling SWF file is in an untrusted local sandbox. <p><i>Permission mechanisms:</i></p> <ul style="list-style-type: none">▪ Policy File; cross-domain access from the local-trust or local-with-networking sandbox requires permission from the website▪ allowScriptAccess—If the request specifies a URL for scripting (such as javascript:), the request fails by default. The containing web page can specify the following permissions using allowScriptAccess:<ul style="list-style-type: none">▪ "never"—not allowed▪ "sameDomain"—allowed if the calling SWF file is from the same sandbox as the container▪ "always"—always allowed from any sandbox.▪ The default for other cross-domain requests is to allow access <p><i>Error notification:</i></p> <ul style="list-style-type: none">▪ SecurityError exception

API Name	Description
MovieClip. getURL()	<p>Opens a browser window to display browser content, or evaluates a browser script.</p> <p><i>Security checks:</i></p> <ul style="list-style-type: none"> ▪ Not allowed if the calling SWF file is in the local-with-file-support sandbox and the resource is nonlocal. <p><i>Permission mechanisms:</i></p> <ul style="list-style-type: none"> ▪ Policy File–Cross-domain access from local-trust or local-with-networking requires permission from the website ▪ allowScriptAccess–If request specifies a URL for scripting (such as javascript:\), the request fails by default. The containing web page can specify permissions using allowScriptAccess: <ul style="list-style-type: none"> ▪ "never"–not allowed ▪ "sameDomain"–allowed if the calling SWF file is from the same sandbox as the container ▪ "always"–always allowed ▪ The default for other cross-domain requests is to allow access. <p><i>Error notification:</i></p> <ul style="list-style-type: none"> ▪ SecurityError exception

Shared object references

Flash applications can store data to disk in local shared objects. Data stored using this mechanism is isolated by domain, so that one domain's locally stored data is not accessible by another. Data is not kept in a predictable location, so it cannot be accessed by client applications other than Flash Player, which enforces the exact domain match rules.

The storage can be configured or reconfigured in various ways by developers, end users, and administrators.

The following table describes the two APIs that can obtain references to shared objects (described in the *Flash Player Security* document). Each API is associated either with an individual SWF file or with some larger part of the sandbox to which it belongs.

Table 6: Shared object APIs

API Name	Description
SharedObject. getLocal() SharedObject. getRemote()	<p>Creates or retrieves a local shared object, or, for <code>getRemote()</code>, an optionally persistent remote shared object.</p> <p>The object is created in or retrieved from a shared object store constrained by the sandbox of the calling SWF file. No errors result from calling <code>getLocal()</code> or <code>getRemote()</code>.</p> <p><i>Security checks:</i></p> <ul style="list-style-type: none"> ▪ There is no mechanism to access shared objects across sandbox boundaries. <p><i>Permission mechanisms:</i></p> <ul style="list-style-type: none"> ▪ Dialog box, Settings Manager, mms.cfg—By default, shared objects can be created up to a maximum of 100K of data per domain. Administrative users and users can place restrictions on the ability to write to the file system. <p><i>Error notification:</i></p> <ul style="list-style-type: none"> ▪ Returns <code>null</code>.

Inter-SWF file (or cross-scripting) APIs

In cross-scripting, ActionScript code from one sandbox can access or manipulate Flash objects from another sandbox. This is again forbidden by default, but can be granted permission using the `System.security.allowDomain()` method. Scripting from the container to Flash Player, such as JavaScript-to-ActionScript scripting, falls into this category as well. The following table describes the APIs that perform script-to-script communication, and includes information about relevant security checks, permission mechanisms, and error notifications:

Table 7: Cross-scripting APIs

API Name	Description
<code>ExternalInterface.addCallback()</code>	<p>Sets up a callback function to be called by a container surrounding Flash Player.</p> <p><i>Security checks:</i></p> <ul style="list-style-type: none"> Cannot overwrite an existing callback created by a SWF file from a different sandbox than calling SWF file. <p><i>Permission mechanisms:</i></p> <ul style="list-style-type: none"> <code>allowDomain</code>—By default, interfaces are associated with the sandbox that called <code>addCallback()</code>; cross-domain access to timers is not allowed unless the <code>allowDomain()</code> method has been invoked. <p><i>Error notification:</i></p> <ul style="list-style-type: none"> <code>SecurityException</code>
<code>LocalConnection.send()</code>	<p>Sends data to a movie clip on the same computer, but not necessarily in the same process.</p> <p><i>Permission mechanisms:</i></p> <ul style="list-style-type: none"> <code>LocalConnection.allowDomain()</code>—By default, <code>LocalConnection</code> objects are associated with the sandbox that created them. Cross-domain access to <code>LocalConnection</code> objects is not allowed unless the <code>LocalConnection.allowDomain()</code> method has been invoked. <p><i>Error notification:</i></p> <ul style="list-style-type: none"> <code>SecurityError</code> event from the <code>LocalConnection</code> object

Outbound scripting

In outbound scripting, ActionScript code from one sandbox calls script code in a container (such as JavaScript in a browser) in another sandbox. Outbound scripting is achieved through use of the `fscommand()` method or the `getURL()` method that specify a scripting statement. This was permitted by default in earlier versions of Flash Player, but it is forbidden by default in Flash Player 8.

If Macromedia Flash content (a SWF file) coming from one domain was embedded in an HTML page from a different domain, the SWF file could read and transfer data, such as cookies, from the HTML server domain to the Macromedia Flash domain. An important security restriction in client-side scripting is that script code is not allowed to inspect, modify, or otherwise interact with any documents that come from a web domain other than the one from which the script itself came. When Flash applications are hosted within HTML pages, they can define and call their own script code using the ActionScript `getURL()` function. When this occurs, originally cross-domain security was not enforced between Flash applications and the HTML pages in which they are hosted. This meant that it was possible to author Flash applications that interact with their surrounding HTML pages, even though the application and its host page may reside in separate domains. This could occur with the ActiveX version of Flash Player 6 (for Internet Explorer) and the plug-in version (for Netscape Navigator). This can only affect websites containing HTML pages that directly source Flash applications that are served from other domains and could be written by individuals not directly trusted by the operator of the website. Examples of this kind of arrangement could include sites that aggregate third-party Flash content and Flash based signatures in message board posts.

For more details on this issue, see *Using AllowScriptAccess to control running of scripts from Flash (TechNote 16494)* or *MPSB02-08-Macromedia Flash Player Cross Server Scripting Security Issue*.

The `allowScriptAccess` parameter of Flash Player was added to Flash Player 6; it controls the ability to perform outbound scripting from within a Flash application, and addresses this potential cross-site scripting security issue. The parameter can have two possible values: `"always"` (outbound scripting always succeeds) and `"never"` (outbound scripting always fails). In earlier versions of Flash Player, the default value is `"always"`; in Flash Player 8, the default value is `"never"` when not specified by the HTML page.

Other sensitive APIs

Privacy is an important aspect of overall security. Macromedia products, including Flash Player, provide very little information that would reveal anything about a user (or their computer). Flash Player does *not* provide personal information about users (such as names, e-mail addresses, and phone numbers), or other sensitive information (such as credit card numbers or account information).

What Flash Player *does* provide is basically standardized hardware and software configuration information that authors might use to enhance the user experiences in the environment encountered. The same information is often available already from the operating system or web browser. The following sections cover the client computer information that Flash Player may make available to Flash applications.

User Agent string

The User Agent string provides very basic information about the platform configuration that the user has on the client machine. It includes the browser, the browser compatibility (particularly important for specialty browsers like AOL or WebTV), the platform, and the operating system, along with their versions where appropriate. The following is an example of a User Agent string:

```
Mozilla/4.0 (compatible; IE 5.5; Windows 98)
```

Every browser also provides all of the information in the User Agent string, so Flash Player is merely making the same information available in a more consistent format. Because different browsers and platforms render HTML content differently, authors can use the User Agent string to deliver the most appropriate content for the best appearance.

Systems capabilities APIs

Authors often want to leverage additional systems capabilities beyond the basic operating system or browser support that the user may have on their platform, such as audio and graphics hardware. Authors can access a wide range of additional information about the multimedia capabilities of the system from the `System.capabilities` object in the `System` class using ActionScript and the system capabilities APIs.

Flash Player 8 can also provide the security configuration settings to ActionScript with the `System` object. Flash Player 8 includes the following additional properties, which, like all `System.capabilities` properties, are read-only:

- `System.capabilities.avHardwareDisable` // Boolean
- `System.capabilities.localFileReadDisable` // Boolean

The following table describes the system capabilities APIs:

Table 8: System capabilities APIs

API Name	Description
<code>System.capabilities.avHardwareDisable</code>	Specifies whether the user's camera and microphone are enabled or disabled. Read-only property.
<code>System.capabilities.hasAccessibility</code>	Indicates whether Flash Player is running on a system that supports communication between Flash Player and accessibility aids. Read-only property.
<code>System.capabilities.hasAudio</code>	Indicates whether Flash Player is running on a system that has audio capabilities. Read-only property.
<code>System.capabilities.hasAudioEncoder</code>	Indicates whether Flash Player is running on a system that can encode an audio stream, such as that coming from a microphone. Read-only property.
<code>System.capabilities.hasEmbeddedVideo</code>	Indicates whether Flash Player is running on a system that supports embedded video. Read-only property.
<code>System.capabilities.hasMP3</code>	Indicates whether Flash Player is running on a system that has an MP3 decoder. Read-only property.
<code>System.capabilities.hasPrinting</code>	Indicates whether Flash Player is running on a system that supports printing. Read-only property.
<code>System.capabilities.hasScreenBroadcast</code>	Indicates whether Flash Player supports the development of screen broadcast applications to be run through Flash Communication Server. Read-only property.
<code>System.capabilities.hasScreenPlayback</code>	Indicates whether Flash Player supports the playback of screen broadcast applications that are being run through Flash Communication Server. Read-only property.
<code>System.capabilities.hasStreamingAudio</code>	Indicates whether Flash Player can play streaming audio. Read-only property.
<code>System.capabilities.hasStreamingVideo</code>	Indicates whether Flash Player can play streaming video. Read-only property.

API Name	Description
<code>System.capabilities.hasVideoEncoder</code>	Indicates whether Flash Player can encode a video stream, such as that coming from a web camera. Read-only property
<code>System.capabilities.isDebugger</code>	Indicates whether Flash Player is an officially released version or a special debugging version. Read-only property
<code>System.capabilities.language</code>	Indicates the language of the system on which Flash Player is running. Read-only property
<code>System.capabilities.localFileReadDisable</code>	Specifies whether Flash Player attempts to read anything (including the first SWF file that Flash Player opens) from the user's hard disk. Read-only property
<code>System.capabilities.manufacturer</code>	Indicates the manufacturer of Flash Player. Read-only property
<code>System.capabilities.os</code>	Indicates the operating system hosting Flash Player. Read-only property
<code>System.capabilities.pixelAspectRatio</code>	Indicates the pixel aspect ratio of the screen. Read-only property
<code>System.capabilities.playerType</code>	Indicates the type of Flash Player: stand-alone, external, plug-in, or ActiveX. Read-only property
<code>System.capabilities.screenColor</code>	Indicates whether the screen is color, grayscale, or black and white. Read-only property.
<code>System.capabilities.screenDPI</code>	Indicates the dots-per-inch screen resolution, in pixels. Read-only property.
<code>System.capabilities.screenResolutionX</code>	Indicates the horizontal size of the screen. Read-only property.
<code>System.capabilities.screenResolutionY</code>	Indicates the vertical size of the screen. Read-only property.
<code>System.capabilities.serverString</code>	A URL-encoded string that specifies values for each <code>System.capabilities</code> property. Read-only property.
<code>System.capabilities.version</code>	A string containing Flash Player version and platform information. Read-only property.

For more details about the API methods and properties that allow authors to access this information, see *ActionScript Language Reference*.

User settings APIs

Flash Player 6 and later allow monitoring microphones and cameras, if they are available on the user's machine. These capabilities can be leveraged in areas such as video conferencing, video and voice chats or messaging, and similar uses. However, to provide user privacy, microphone and video access is allowable only when the user approves it, and the user may disable it at any time. By default, Flash Player prompts the user whenever the use of such devices is requested by a Flash application. The user can also adjust their settings (using the Settings Manager) to always deny, allow, or prompt for such use.

Camera

An interface lets the user set persistent privacy settings per domain that control access by the Flash application to the system camera. The default is to deny access. The following table describes the API that provides many additional aspects of the camera attributes and operations that are available to the author:

Table 9: Camera security-related API

API Name	Description
<code>Camera.get()</code>	Returns a default or specified Camera object. <i>Permission Mechanism:</i> User dialog box, Settings Manager mms.cfg. By default, Flash Player displays a Privacy dialog box that lets the user choose to allow or deny access to the camera. End users and administrative users may also disable camera access on a per-site or global basis. <i>Error notification:</i> Returns <code>null</code> if the camera is not available.

Audio In (Microphone)

Flash Player allows the user to set persistent privacy settings per domain (using the Settings Manager or other user interface prompts), including access by Flash applications to the system microphone. The default is to deny access. The following table describes the microphone-related API, and includes information about relevant permissions mechanisms and error notifications:

Table 10: Microphone security-related API

API Name	Description
<code>Microphone.get()</code>	Returns a default or specified Microphone object. <i>Permission Mechanism:</i> User dialog, Settings Manager mms.cfg-By default, Flash Player displays a Privacy dialog box that lets the user choose to allow or deny access to the microphone. End users and Administrative users may also disable camera access on a per-site or global basis. <i>Error notification:</i> Returns <code>null</code> if the microphone is not available.

Clipboard

The `System.setClipboard` API allows a SWF file to replace the contents of the clipboard with a plain-text string of characters or allows clearing the clipboard. (To protect against the risk posed by passwords and other sensitive data often being cut or copied to clipboards, there is no corresponding “getClipboard” (read) function.) The following table describes the clipboard API:

Table 11: Clipboard API

API Name	Description
<code>System.setClipboard()</code>	Replaces the contents of the clipboard with a specified text string Write-only property.

Mouse and keyboard APIs

Flash Player 5 and later provide the event model that allows the author to handle events in a Flash application, such as mouse clicks and keyboard input. This model also includes events such as tab order, selections, text field, and more sophisticated data events, and it allows access of these events as methods.

A Flash application can only monitor keyboard and mouse events that occur within its focus. A Flash application cannot detect keyboard or mouse events in another application.

The following table describes the various supported keyboard and mouse APIs, even though there are few security-related aspects associated with these events:

Table 12: Keyboard and mouse APIs

API Name	Description
<code>Mouse.addListener()</code>	Registers an object to receive <code>onMouseDown</code> , <code>onMouseMove</code> , <code>onMouseWheel</code> , and <code>onMouseUp</code> notification.
<code>Mouse.hide()</code>	Hides the mouse pointer in the SWF file.
<code>Mouse.removeListener()</code>	Removes an object that was registered with <code>addListener()</code> .
<code>Mouse.show()</code>	Displays the mouse pointer in the SWF file.
<code>Mouse.onMouseDown</code>	Listener that notifies when the mouse button is pressed.
<code>Mouse.onMouseMove</code>	Listener that notifies when the mouse is moved.
<code>Mouse.onMouseUp</code>	Listener that notifies when the mouse button is released.
<code>Mouse.onMouseWheel</code>	Listener that notifies when the user rolls the mouse wheel.
<code>Key.addListener()</code>	Registers an object to receive notification when the <code>onKeyDown()</code> and <code>onKeyUp()</code> methods are invoked.
<code>Key.getAscii()</code>	Returns the ASCII value of the last key pressed.
<code>Key.getCode()</code>	Returns the virtual key code of the last key pressed.
<code>Key.isDown()</code>	Returns <code>true</code> if the key specified in the parameter is pressed.
<code>Key.isToggled()</code>	Returns <code>true</code> if the Num Lock or Caps Lock key is pressed.
<code>Key.removeListener()</code>	Removes an object that was previously registered with <code>Key.addListener()</code> . <i>Error notification:</i> Returns <code>false</code> if the listener was not successfully removed.
<code>Key.onKeyDown()</code>	Listener that notifies when a key is pressed.
<code>Key.onKeyUp()</code>	Listener that notifies when a key is released.

Flash Player updates

Flash Player runs as a virtual machine, and all content executed by Flash Player runs inside that virtual machine. However, there are methods available to execute specific native binary programs used to update Flash Player and to provide runtime add-ins to the player. These methods are not available for use by most developers, but are included here for completeness. Flash Player provides limitations on access to these update mechanisms, as the following table describes:

Table 13: Native binary execution

Operation	Description
<code>FlashPlayer.autoUpdate</code>	<p>Allows Flash Player to update itself to the latest version.</p> <p><i>Security check:</i> Auto-update data is PKCS-7 signed (a public key cryptography standard) using a Macromedia private key.</p> <p><i>Permission mechanisms:</i> mms.cfg, Settings Manager, user prompt at installation time.</p>
<code>System.product()</code>	<p>Allows Flash Player to install add-ins or native capabilities from trusted sources.</p> <p><i>Security check:</i> Auto-update data is PKCS-7 signed using a Macromedia private key.</p> <p><i>Permission mechanisms:</i> User prompted at installation time.</p>