

**Adobe® Flash® Access™**

August 2011

Version 3.0

# Using the Flash Access Reference Implementations



© 2010 Adobe Systems Incorporated. All rights reserved.

Using the Flash Access Reference Implementations

This guide is protected under copyright law, furnished for informational use only, is subject to change without notice, and should not be construed as a commitment by Adobe Systems Incorporated. Adobe Systems Incorporated assumes no responsibility or liability for any errors or inaccuracies that may appear in the informational content contained in this guide.

This guide is licensed for use under the terms of the Creative Commons Attribution Non-Commercial 3.0 License. This License allows users to copy, distribute, and transmit the user guide for noncommercial purposes only so long as (1) proper attribution to Adobe is given as the owner of the user guide; and (2) any reuse or distribution of the user guide contains a notice that use of the user guide is governed by these terms. The best way to provide notice is to include the following link. To view a copy of this license, visit <http://creativecommons.org/licenses/by-nc-sa/3.0/>

Adobe, the Adobe logo, Adobe AIR, Flash Access, Flash Player, and Flex are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States and/or other countries.

Apple and Mac OS are trademarks of Apple Inc., registered in the United States and other countries. Java is a trademark or registered trademark of Sun Microsystems, Inc. in the United States and other countries. Linux is the registered trademark of Linus Torvalds in the U.S. and other countries. Microsoft and Windows 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.

Updated Information/Additional Third Party Code Information available at <http://www.adobe.com/go/thirdparty>.

Portions include software under the following terms:

This product contains either BSAFE and/or TIPEM software by RSA Security Inc.

Adobe Systems Incorporated, 345 Park Avenue, San Jose, California 95110, USA.

Notice to U.S. Government End Users. The Software and Documentation are "Commercial Items," as that term is defined at 48 C.F.R. §2.101, consisting of "Commercial Computer Software" and "Commercial Computer Software Documentation," as such terms are used in 48 C.F.R. §12.212 or 48 C.F.R. §227.7202, as applicable. Consistent with 48 C.F.R. §12.212 or 48 C.F.R. §§227.7202-1 through 227.7202-4, as applicable, the Commercial Computer Software and Commercial Computer Software Documentation are being licensed to U.S. Government end users (a) only as Commercial Items and (b) with only those rights as are granted to all other end users pursuant to the terms and conditions herein. Unpublished-rights reserved under the copyright laws of the United States. Adobe Systems Incorporated, 345 Park Avenue, San Jose, CA 95110-2704, USA. For U.S. Government End Users, Adobe agrees to comply with all applicable equal opportunity laws including, if appropriate, the provisions of Executive Order 11246, as amended, Section 402 of the Vietnam Era Veterans Readjustment Assistance Act of 1974 (38 USC 4212), and Section 503 of the Rehabilitation Act of 1973, as amended, and the regulations at 41 CFR Parts 60-1 through 60-60, 60-250, and 60-741. The affirmative action clause and regulations contained in the preceding sentence shall be incorporated by reference.

# Contents

## **Using the reference implementations**

Command line tools for packaging content and creating revocation lists .....	1
License server and watched folder packager .....	15
Adobe Flash Access Manager AIR application usage .....	27

# Using the reference implementations

The Flash Access product comes with a reference implementation for the following components:

- Command line tools for packaging content and creating revocation lists
- License server and watched folder packager
- Adobe Flash Access Manager AIR application

*Note: You should deploy either the Flash Access Server for Protected Streaming, the reference implementation, or your own license server.*

## Command line tools for packaging content and creating revocation lists

The reference implementation includes the following command line tools:

- Policy Manager: A tool for creating and managing policies
- Media Packager: A tool for creating encrypted FLV and F4V files
- Policy Update List Manager: A tool for creating and viewing policy update lists
- Revocation List Manager: A tool for creating and viewing revocation lists
- AIR Publisher ID Utility License Generator (Flash Access Professional only) License Embedder (Flash Access Professional only)

## Requirements

The requirements for using the command line tools available in the reference implementations are as follows:

- All of the command line tools require Java 1.5 or higher.
- Packager and License Server credentials (certificate and password) that are issued by Adobe. You need credentials to encrypt and sign video files, to sign Policy Update and Revocation lists, and to pre-generate licenses.
- A 32-bit operating system. The tools are not officially supported on 64-bit operating systems (although they may work).

*Note: Because of a Java bug, arguments that are used on the command line, such as file names or policy names or descriptions, must use characters only from the operating system's default character set.*

## Configuration file

Several of the command-line tools require a configuration file that contains information for the tools to use to apply policies and encrypt files.

The default configuration file is flashaccesstools.properties and is located in the working directory; that is, the directory from which you run the tools (see Installing the command line tools). Each tool also contains an option (-c) that lets you point to the configuration file you want to use if you prefer not to use the default.

The configuration file uses the Java property file format. If values for any of the properties contain special characters, keep in mind the following restrictions:

- Escape backslashes with an additional backslash. For example, to specify the C:\credentials.pfx file, specify it as C:\\credentials.pfx or C:/credentials.pfx. To specify a file on a network server, specify \\server\folder\filename.pfx.
- The configuration file can contain only Latin-1 characters. If you must use non-Latin-1 characters, use the appropriate Unicode escape sequence (using, optionally, the native2ascii tool that comes with Java).

Set values for properties in the configuration file before you run the tools. For some of the command line tools, you can set the values for some options through either the command line or the configuration file. In those cases, values that are set through the command line take precedence over any values in the configuration file.

## Installing the command line tools

You can copy the files you need from the \Reference Implementation\Command Line Tools directory on the DVD, which contains the default flashaccesstools.properties configuration file, and a libs directory, which contains the JAR files for the tools. The samples directory contains several sample Java source files demonstrating usage of the Flash Access SDK APIs. To build and run the samples, use the build-samples.xml Ant script.

## Policy Manager

Using Policy Manager, you can create and manage policies. Before you run Policy Manager, optionally set values for Policy Manager properties in the configuration file. The configuration file specifies information that will be applied to all policies. All Policy Manager properties may also be specified on the command line.

## Configuration file properties

The configuration file specifies the following properties. For property names that include *n*, *n* represents an integer starting with 1 and increasing for each instance of the property.

Property/Command Line Option	Description
policy.name -n <i>policyname</i>	The human-readable policy name.
policy.critical -critical <i>boolean</i>	<b>New in 3.0.</b> Set policy criticality. If true, the server must understand all parts of the policy (this is the default behavior). If false, the server may ignore policy attributes it does not understand.
policy.chaining.asymmetric.certfile	<b>New in 3.0.</b> License server certificate whose public key is used to encrypt the root encryption key for the Enhanced License Chaining. This property specifies a file that contains the certificate only (either PEM or DER format is acceptable).
policy.chaining.rootKey -rootKey <i>root-key</i>	<b>New in 3.0.</b> Specify root encryption key for the Enhanced License Chaining. If no key is specified, and Enhanced License Chaining is enabled, a random key will be generated. The key must be 16 bytes in length and specified as Hex values. Whitespace between the Hex values is optional. The command line option is not allowed for updates.

Property/Command Line Option	Description
<p>policy.domain.url</p> <p>-domainURL <i>url</i></p>	<p><b>New in 3.0.</b></p> <p>URL of domain server, if domain registration is required. The command line option is not allowed for updates.</p>
<p>policy.domain.anonymous</p> <p>-domainAnon</p>	<p><b>New in 3.0.</b></p> <p>Specifies whether anonymous domain registration is allowed. Set the property to true or include this command line option to allow anonymous access. The command line option is not allowed for updates and cannot be used with -domainAuthNS.</p>
<p>policy.domain.authNamespace</p> <p>-domainAuthNS <i>namespace</i></p>	<p><b>New in 3.0.</b></p> <p>The authentication namespace for domain registration. If specified, the client should authenticate with a user name and password issued by the specified authority. This command line option cannot be used with -domainAnon, and it is not allowed for updates.</p>
<p>policy.outputProtection.analog</p> <p>-opAnalog <i>AnalogOption</i></p>	<p>Analog output protection constraints. In Flash Access 3.0, the following values are supported:</p> <ul style="list-style-type: none"> <li>• NO_PROTECTION</li> <li>• USE_IF_AVAILABLE</li> <li>• USE_IF_AVAILABLE_ACP</li> <li>• USE_IF_AVAILABLE_CGMSA</li> <li>• REQUIRED</li> <li>• REQUIRED_ACP</li> <li>• REQUIRED_CGMSA</li> <li>• NO_PLAYBACK</li> </ul>
<p>policy.drmVersionBlacklist.n</p> <p>-drmBlacklist <i>name/value-pairs</i></p>	<p>DRM clients restricted from accessing protected content. In Flash Access 3.0, the following name/value pairs are valid:</p> <p><code>os   release   arch   model   vendor   env   screen=value</code></p> <p>A list of versions of DRM modules that may not be used (black list). The property must use the following format: <code>os : stringValue:release : stringValueAdditional name/value pairs must be comma-separated.</code></p>
<p>policy.runtimeVersionBlacklist.n</p> <p>-runtimeBlacklist <i>name/value-pairs</i></p>	<p>Application runtimes restricted from accessing protected content. In Flash Access 3.0, the following name/value pairs are valid:</p> <p><code>os   release   application   arch   model   vendor   env   screen=value</code></p> <p>A list of versions of runtime modules that may not be used (black list). The property must use the following format: <code>os : stringValue:application : stringValue:release : stringValueAdditional name/value pairs must be comma-separated.</code></p>
<p>policy.v1DeviceCapabilities</p> <p>-devCapabilitiesV1 <i>name/value-pairs</i></p>	<p><b>New in 2.0.2.</b></p> <p>Specifies device capabilities required to access protected content. The value consists of comma separated name=value pairs with the following format:</p> <p><code>nonUserAccessibleBus   hardwareRootOfTrust=true   false</code></p> <p>For example, "nonUserAccessibleBus=false". During update, use -devCapabilitiesV1 without the remaining arguments to remove the device capabilities restriction.</p>

Property/Command Line Option	Description
<p>policy.syncFrequency</p> <p>-sync <i>name/value-pairs</i></p>	<p><b>New in 3.0.</b></p> <p>Specify how often clients are required to send synchronization messages to the server. If not set, clients will not send synchronization messages when playing content protected with this policy. The value consists of comma separated name=value pairs with the following format:</p> <pre>start   force   hardStop=numberValue</pre> <p>Start interval("start"), required, specifies the client should start synchronizing with the server this many minutes since the last synchronization. Force synchronization probability("force"), optional, is the probability (0-100) with which the client should force a synchronization message during playback. Hard stop interval("hardStop"), optional, is the time in minutes after which the client will fail playback if unable to synchronize. If set, must be greater than start interval. During update, use -sync without the remaining arguments to remove the synchronization requirements.</p>
policy.useRootLicense	Indicates whether this policy has a root license (see <i>Enhanced License Chaining</i> in <i>Using Flash Access Server for Protecting Content</i> ).
policy.startDate	The date after which content is valid. Use the format <i>yyyy-mm-dd</i> (for example, 2009-01-31 represents January 31 at 12:00 AM) or <i>yyyy-mm-dd-h24:min:sec</i> (for example, 2009-01-31-14:30:00 represents January 31 at 2:30 PM).
policy.expiration.endDate	The date before which content is valid. Both policy.expiration.endDate and policy.expiration.duration may not be specified concurrently. Use the format <i>yyyy-mm-dd</i> or <i>yyyy-mm-dd-h24:min:sec</i> (for example, 2009-01-31-14:30:00 represents January 31 at 2:30 PM).
policy.expiration.duration	The amount of time the content is valid (in minutes), starting from when it is packaged. Both policy.expiration.endDate and policy.expiration.duration may not be specified at the same time.
policy.licenseCaching.duration	Amount of time a license may be cached on the client (in minutes). Set this property to 0 to disallow license caching. The value must be 0 or higher. Both policy.licenseCaching.duration and policy.licenseCaching.endDate may not be used concurrently.
policy.licenseCaching.endDate	The date after which licenses may not be cached. Both policy.licenseCaching.duration and policy.licenseCaching.endDate may not be used concurrently.
policy.anonymous	Indicates whether anonymous license acquisition is allowed. The default is "false" (username/password authentication is required) if not specified.
policy.authNamespace	If username/password authentication is required, this property specifies an optional name qualifier for user names.
policy.customProp.n	Custom name/value pairs to be used by the server during license acquisition. Use the following format for specifying properties: policy.customProp.n= <i>name=value</i>
policy.playbackWindow	Specifies the playback window (in minutes), which is the duration for which the license is valid after the first time it is used to play protected content.
policy.outputProtection.digital	Output protection constraints. Values must be one of the following: <i>NO_PROTECTION, USE_IF_AVAILABLE, REQUIRED, NO_PLAYBACK</i>
policy.drmMinSecurityLevel	The DRM module must have the specified minimum security level, or higher, to access protected content.
policy.runtimeMinSecurityLevel	The application runtime module must have the specified minimum security level, or higher, to access protected content.

Property/Command Line Option	Description
policy.allowedAIRApplication.n	A white list of Adobe AIR applications allowed to play protected content. The property must use the following format: <code>pubId[:appId[:[min]:[max]]]</code>
policy.allowedSWFApplication.n	A white list of SWF applications allowed to play protected content. Use the following format:  <code>URL</code> or <code>file=swf_file,time=max_time_to_verify</code> <code>swf_file</code> is the SWF file for which to compute the hash and <code>max_time_to_verify</code> is the maximum time to allow for download and verification of the SWF to complete (in seconds).
policy.license.customProp.n	Custom name/value pairs to be included in licenses issued to users. Use the following format:  <code>policy.license.customProp.n=name=value</code>  This option can be defined multiple times for multiple custom properties.

### Command line usage

Before using Policy Manager, ensure that you fulfill the requirements listed in Requirements.

Policy Manager is in the \Reference Implementation\Command Line Tools directory on the DVD. To run the tool, use the following syntax:

```
java -jar AdobePolicyManager.jar command filename [options]
```

The following table contains descriptions of the command line actions shown in the syntax above:

Command line action	Description
new	Creates a new policy.
detail	Describes an existing policy.
update	Updates an existing policy.

The following table describes the command line options that can be specified along with the syntax above:

Command line option	Description
<code>-c configfile</code>	Specify the location of the configuration file. If this option is not used, the Policy Manager will look for <code>flashaccesstools.properties</code> in the working directory. Options specified on the command line take precedence over those present in the configuration file.
<code>-o</code>	If the destination file already exists, overwrite it without prompting.
<code>-noprompt</code>	Do not ask if the destination file should be overwritten. If the destination file already exists and <code>-o</code> is not set, an error will be returned.
<code>-root</code>	Indicates the policy has a root license. Not allowed for updates.
<code>-e date</code>	The date before which licenses will be valid. Specify as <code>yyyy-mm-dd</code> or <code>yyyy-mm-dd-h24:min:sec</code> . For example, <code>2008-12-1</code> or <code>2008-12-1-00:00:00</code> for midnight on December 1, 2008. The value must be greater than the value of <code>-s</code> , if present. This option cannot be used with <code>-r</code> . To remove the end date when updating a policy, use <code>-e</code> without specifying a date.

Command line option	Description
<code>-r minutes</code>	The duration (minutes) that content protected with this policy is valid, beginning when the content is protected with the packager. The value must be non-negative. This option cannot be used with <code>-e</code> . To remove the duration when updating a policy, use <code>-r</code> without specifying a number of minutes.
<code>-s date</code>	The date after which licenses will be valid. Specify as <code>yyyy-mm-dd</code> or <code>yyyy-mm-dd-h24:min:sec</code> . For example, 2008-12-1 or 2008-12-1-00:00:00 for midnight on December 1, 2008. The value must be less than the value of <code>-e</code> , if present. This option cannot be used with <code>-r</code> . To remove the start date when updating a policy, use <code>-s</code> without specifying a date.
<code>-w minutes</code>	The playback window (the number of minutes the content may be viewed, beginning from the first playback). If this option is not specified or if <code>-w</code> is used without specifying the number of minutes, there is no playback window limitation. The value must be non-negative.
<code>-l minutes</code>	The license caching duration in minutes, which is the time a license will be allowed to be cached in the client's License Store after the license has been issued by the server. The value must be non-negative. Specify <code>-l 0</code> to indicate license caching is not permitted. Use <code>-l</code> without specifying a number of minutes for unlimited license caching.
<code>-ldate date</code>	The license caching end date (the date after which licenses may not be cached in the client's License Store, after the license has been issued by the server). Specify as <code>yyyy-mm-dd</code> or <code>yyyy-mm-dd-h24:min:sec</code> . For example, 2008-12-1 or 2008-12-1-00:00:00 for midnight on December 1, 2008. Use <code>-l</code> without specifying a number of minutes for unlimited license caching.
<code>-authNS</code>	The authentication namespace. If specified, the client should authenticate with a user name and password issued by the specified authority. This option cannot be used with <code>-x</code> . It is not allowed for updates.
<code>-x</code>	Allow anonymous access. This option cannot be used with <code>-authNS</code> . It is not allowed for updates.
<code>-air pubId[:appId[:[min]:[max]]]</code>	A whitelist of AIR applications allowed to play protected content. Use this to restrict which publishers, applications, and versions may access content protected with this policy.  If <code>appId</code> is not specified, all applications for publisher <code>pubId</code> are allowed.  <code>min</code> and <code>max</code> version numbers are optional.  Multiple <code>-air</code> options may be specified to allow multiple applications. If no AIR or SWF applications are specified, all applications may access this content. During an update, use <code>-air</code> without the remaining arguments to remove all entries from the list.
<code>-drmBlacklist name/valuepairs</code>	The DRM clients restricted from accessing protected content. The value consists of comma separated name:value pairs with the following format:  <code>os   release=stringValue</code>  For example, 'os=Win,release=2.0.1'. During an update, use <code>-drmBlacklist</code> without the remaining arguments to remove all entries from the list.

Command line option	Description
-drmLevel <i>int</i>	Indicates that DRM clients must have the specified minimum security level to access protected content.
-opAnalog NO_PROTECTION   USE_IF_AVAILABLE   REQUIRED   NO_PLAYBACK   ACP_REQUIRED   CGMS-A_REQUIRED   USE_ACP_IF_AVAILABLE   USE_CGMS-A_IF_AVAILABLE	Analog output protection constraints.
-opDigital NO_PROTECTION   USE_IF_AVAILABLE   REQUIRED   NO_PLAYBACK	Digital output protection constraints.
-runtimeBlacklist <i>name/valuepairs</i>	<p>The application runtimes restricted from accessing protected content. The value consists of comma separated name:value pairs with the following format:</p> <p><b>New values supported in 3.0.</b></p> <p><i>os   application   release=stringValue</i></p> <p>For example, 'os=Win,release=2.0.1,application=AIR'. During an update, use -runtimeBlacklist without the remaining arguments to remove all entries from the list.</p>
-runtimeLevel <i>int</i>	Indicates that the application runtimes must have the specified minimum security level to access protected content.
-swf <i>url</i> -swf file= <i>swf_file</i> , time= <i>max_time_to_verify</i>	A whitelist of SWF applications allowed to play protected content. Multiple -swf options may be specified to allow multiple applications. If no AIR or SWF applications are specified, all applications may access this content. During an update, use -swf without the remaining arguments to remove all entries from the list. To identify a SWF by its hash value, specify the SWF file for which to compute the hash and the maximum time to allow for SWF verification to complete (in seconds).
-k <i>name=value</i>	Specifies custom key/values to add to the policy. Multiple -k options may be specified. During update, use -k without the remaining arguments to remove all properties. The interpretation or handling of this data is completely up to the implementation of the Flash Access license server.
-p <i>name=value</i>	Adds a custom property, which will appear in the license generated for each client. Multiple -p options may be specified to add multiple properties. During an update, use -p without the remaining arguments to remove all properties. The interpretation or handling of this data is completely up to the implementation of the client application.

## Media Packager

Using Media Packager, you can specify what data in the file to encrypt and the policy to apply to the content file. For example, you can specify that the video data is encrypted but the audio data is unencrypted.

### Configuration file properties

Before you run Media Packager, specify values for the Media Packager properties. The configuration file specifies the following properties. For property names that include *n*, *n* represents an integer starting with 1 and increasing for each instance of the property.

Property	Description
encrypt.contents.video	Indicates whether to encrypt video content.
encrypt.contents.audio	Indicates whether to encrypt audio.
encrypt.contents.script	Indicates whether to encrypt script data in FLVs. <i>onMetaData</i> and <i>onXMP</i> script data tags are never encrypted, even if this option is enabled.
encrypt.contents.video.level	Indicates the video encryption level. A value of high is used to encrypt all video content, while values of medium and low are used to encrypt portions of the video content for F4V files containing H.264 content.  value = high   medium   low
encrypt.contents.secondsUnencrypted	If the value is greater than 0, the specified number of seconds of content at the beginning of the file will not be encrypted.
encrypt.keys.asymmetric.certfile	The license server certificate file used to encrypt the key. The <code>encrypt.keys.asymmetric.certfile</code> property specifies a file that contains the certificate only (either PEM or DER format is acceptable).
encrypt.keys.policyFile.n	This property is used repeatedly to create a list of policies to apply to the content. n is an integer whose value is 1 or greater. The client will use the first instance by default.
encrypt.license.serverurl	The license server URL.
encrypt.license.servercert	The transport certificate for the license server. This property specifies a .cer file that contains the certificate only (either PEM or DER format is acceptable).
encrypt.sign.certfile	The PKCS12 file containing packager credentials for signing content. The <code>encrypt.sign.certfile</code> should refer to a .pfx file containing a certificate and private key.
encrypt.sign.certpass	The password used to protect the file specified by <code>encrypt.sign.certfile</code> .
encrypt.keys.policyFile.n.domain.transportcert	<b>New in 3.0.</b>  If a policy <code>encrypt.keys.policyFile.n</code> requires domain registration with a server that uses a different transport certificate than specified in <code>encrypt.license.servercert</code> , the domain transport certificate needs to be provided.  This property specifies a file that contains the certificate only (either PEM or DER format is acceptable).
encrypt.keys.licenseKey	<b>New in 3.0.</b>  Specify license key. If no key is specified, the key will be randomly generated. When key rotation is not enabled, this is the key used to encrypt the content.  When key rotation is enabled, this key is used to protect the rotation keys. The key must be 16 bytes in length and specified as Hex values. Whitespace between the Hex values is optional.
encrypt.keys.rotation.enable	<b>New in 3.0.</b>  Specifies whether key rotation is enabled. If set to false (default), key rotation is disabled and the master CEK will be used to encrypt all samples in the content.  If set to true, key rotation will be enabled, and different keys can be used to encrypt portions of the content.

Property	Description
encrypt.keys.rotation.key.n	<b>New in 3.0.</b> Sequence of rotated keys used to encrypt content when key rotation is enabled. If no keys are specified, keys will be randomly generated. The keys must be 16 bytes in length and specified as Hex values.  Whitespace between the Hex values is optional. n must be monotonically increasing, starting from 1. When multiple keys are specified, the keys will be cycled through in the order indicated.
encrypt.keys.rotation.interval	<b>New in 3.0.</b> Specifies the interval (in seconds) during which a rotation key will be used to encrypt content samples.  After this amount of time in the content has been encrypted, the next rotation key will be used. If key rotation is enabled and no interval is specified, keys will be rotated every 15 minutes.
encrypt.license.serverless	<b>New in 3.0.</b> If true, there is no license server from which the licenses can be obtained. Licenses must be embedded or obtained out-of-band. Default is false if not specified. Only supported in Flash Access Professional.

## Command line usage

Before using Media Packager, ensure that you fulfill the requirements listed in Requirements and that the configuration file contains the required information (see Configuration file).

Media Packager is in the \Reference Implementation\Command Line tools directory on the DVD. To encrypt a single file, use the following syntax:

```
java -jar AdobePackager.jar source dest [options]
```

- *source* is the file to be encrypted.
- *dest* specifies where the encrypted content will be written. If a directory is specified, the encrypted file will be saved in this folder using the same file name as the source file, but the directory must not be the directory which contains the source file.

To encrypt multiple files with the same key (for multi-bit-rate support), use the following syntax:

```
java -jar AdobePackager.jar sourcefiles dest-directory [options]
```

- *sourcefiles* is a series of whitespace-delimited source entries representing the files to be encrypted.
- *dest-directory* specifies where the encrypted content will be written. The encrypted files will be saved in this folder using the same file names as the source files, but the directory must not be the directory that contains the source files.

To view information about an encrypted file, use the following syntax:

```
java -jar AdobePackager.jar -d encryptedfile [-e] [-m]
```

- *encryptedfile* is the encrypted file.

**Note:** During packaging, the Media Packager will no longer generate a .header file by default. To generate this file, use the -h option during packaging.

The following table contains descriptions of the command line options shown in the syntax above:

Command line option	Description
<code>-c <i>configfile</i></code>	Specifies the location of the configuration file. If this option is not used the Media Packager will look for flashaccesstools.properties in the working directory.
<code>-d <i>encryptedfile</i></code>	Shows information about a file that was already packaged. The source and destination files are not required.
<code>-e</code>	Use this option with <code>-d</code> to extract policies from a packaged file. A file will be created in the same directory as the encrypted file using the file name and policy identifier.
<code>-h</code>	Use with <code>-d</code> to extract the DRM header from a packaged file. A file is created in the same directory as the encrypted file, using the file name and the extension '.header'
<code>-i <i>contentID</i></code>	Specifies a unique identifier for this piece of content. If no identifier is specified, the destfile file name will be used.
<code>-k <i>key= value</i></code>	Specifies a custom key/value to add to content metadata. Multiple <code>-k</code> options may be specified.
<code>-m</code>	Use this option with <code>-d</code> to extract metadata from a packaged file. A file will be created in the same directory as the encrypted file using the file name and the extension ".metadata".
<code>-noprompt</code>	Do not ask whether the destination file should be overwritten. If the destination file already exists and <code>-o</code> is not set, an error will be returned.
<code>-o</code>	Overwrites the destination file without prompting, if it already exists.
<code>-p <i>filename [domain-transport-cert]</i></code>	<p><b>Updated in 3.0.</b></p> <p>The optional second parameter is supported from Flash Access 3.0.</p> <p>Specifies the name of the file containing the policy. If the policy requires domain registration with a server that uses a different transport certificate than the one specified in the properties file, the domain transport certificate also needs to be provided.</p> <p>Multiple <code>-p</code> options may be specified, and the client will use the first by default. The values specified on the command line take precedence over those specified in the configuration file.</p>

## Policy Update List Manager

Before using Policy Update List Manager, ensure that you fulfill the requirements listed in Requirements and that the configuration file contains the required information (see Configuration file).

### Configuration file properties

The following are the Policy Update List Manager properties, which specify a PKCS12 file containing credentials for signing revocation lists (License Server Certificate):

```
revocation.sign.certfile=license-server-credentials.pfxrevocation.sign.certpass=password
```

### Command line usage

Policy Update List Manager is in the \Reference Implementation\Command Line Tools directory on the DVD. To create a policy update list, use the following syntax:

```
java -jar AdobePolicyUpdateListManager.jar destfile [options]
```

- *destfile* indicates where the policy update list will be written.

To view an existing policy update list, use the following syntax:

```
java -jar AdobePolicyUpdateListManager.jar -d filename
```

The following table contains descriptions of the command line options shown in the syntax above:

Command line option	Description
<code>-c configfile</code>	Specifies the location of the configuration file. If this option is not used, the Policy Update List Manager will look for flashaccesstools.properties in the working directory.
<code>-d filename</code>	Displays information about the policy update list.
<code>-e date</code>	(Optional) The expiration date of the policy update list. Use the format <code>yyyy-mm-dd</code> or <code>yyyy-mm-dd-h24:min:sec</code> (for example, 2009-01-31-14:30:00 represents January 31 at 2:30 PM).
<code>-f filename [certfile]</code>	<b>Updated in 3.0.</b> The optional second parameter is supported from Flash Access 3.0. Adds all entries from the existing policy update list. Only one existing file may be specified. If this existing list was signed with a different credential than the one being used to sign the new list, specify its certificate file, so its signature can be verified.
<code>-noprompt</code>	Do not ask if the destination file should be overwritten. If the destination file already exists and <code>-o</code> is not set, an error will be returned.
<code>-o</code>	If the destination file already exists, overwrite it without prompting.
<code>-r policyIDdate "reasonCode" "reasonText" "reasonURL"</code>	(Optional) Revokes the policy ID on the specified date. An optional reason code, reason text, and reason URL may also be provided. Specify an empty string "" to indicate that no value is provided for the optional parameters. Specify the date as <code>yyyy-mm-dd</code> or <code>yyyy-mm-dd-h24:min:sec</code> (for example 2008-12-1 or 2008-12-1-00:00:00 for midnight on December 1, 2008). If a date is not specified, the current date is used. The reason code must be greater than or equal to 0. Multiple <code>-r</code> options may be specified.
<code>-rf policyFilename date "reasonCode" "reasonText" "reasonURL"</code>	Performs the same action as the <code>-r</code> flag, but extracts the policy identifier from the given file.
<code>-u policyFilename "reasonCode" "reasonText" "reasonURL"</code>	Replaces any matching policy in a license request with this policy using the given reason code (optional), reason text (optional), and reason URL (optional). Specify an empty string "" to indicate that no value is provided for the optional parameters. The reason code must be greater than or equal to 0. Multiple <code>-u</code> options may be specified.

## Revocation List Manager

Before using Revocation List Manager, ensure that you fulfill the requirements listed in Requirements and that the configuration file contains the required information (see Configuration file).

### Configuration file properties

The following are the Revocation List Manager properties, which specify a PKCS12 file containing credentials for signing revocation lists (License Server Certificate):

```
revocation.sign.certfile=license-server-credentials.pfxrevocation.sign.certpass=password
```

## Command line usage

Revocation List Manager is in the \Reference Implementation\Command Line Tools directory on the DVD. To run the tool, use one of the following syntaxes:

```
java -jar AdobeRevocationListManager.jar destfilecrlNumber [options]
java -jar AdobeRevocationListManager.jar -d filename
```

- *destfile* indicates where the revocation list will be written.
- *crlNumber* is a non-negative version number of the CRL. This number should be incremented each time the CRL is updated.

The following table contains descriptions of the command line options shown in the syntax above:

Command line option	Description
<i>-c configfile</i>	Specifies the location of the configuration file. If this option is not used, the Revocation List Manager will look for flashaccesstools.properties in the working directory.
<i>-d filename</i>	Displays information about the revocation list.
<i>-e date</i>	(Optional) The expiration date of the revocation list. Use the format <i>yyyy-mm-dd</i> or <i>yyyy-mm-dd-h24:min:sec</i> (for example, 2009-01-31-14:30:00 represents January 31 at 2:30 PM).
<i>-f filename [certfile]</i>	<b>Updated in 3.0.</b> The optional second parameter is supported from Flash Access 3.0. Adds all entries from the existing revocation list. Only one existing file may be specified.  If this existing list was signed with a different credential than the one being used to sign the new list, specify its certificate file next, so its signature can be verified.
<i>-noprompt</i>	Do not ask if the destination file should be overwritten. If the destination file already exists and <i>-o</i> is not set, an error will be returned.
<i>-o</i>	If the destination file already exists, overwrite it without prompting.
<i>-r issuerName serialNumber revocation Date</i>	Revokes the certificate identified by issuerName and serialNumber on the given date. The issuerName must follow the 509 name format (for example, "CN=12345,O=Adobe Systems Incorporated,C=US"). Specify serial numbers in hexadecimal form. Specify the revocation date as <i>yyyy-mm-dd</i> or <i>yyyy-mm-dd-h24:min:sec</i> , for example 2008-12-1 or 2008-12-1-00:00:00 for midnight on December 1, 2008. If the revocation date is not specified, the current date is used.

## AIR Publisher ID utility

As part of the process of building an AIR file, the AIR Developer Tool (ADT) generates a Publisher ID. This is an identifier that is unique to the certificate used to build the AIR file. If you reuse the same certificate for multiple AIR applications, they will have the same Publisher ID. The AIR Publisher ID utility is used to compute the Publisher ID for an AIR application. AIR releases after 1.5.2 do not write the generated Publisher ID to a file, so it is necessary to use this tool to determine the Publisher ID if you are using an AIR application whitelist.

**Note:** The Publisher ID, used for AIR whitelist enforcement is not the same as the Publisher ID, which the application publisher can specify in the application's application.xml file.

## Command line usage

To run the tool, use the following syntax:

```
java -jar AdobePublisherIDUtility.jar signaturefile
java -jar AdobePublisherIDUtility.jar -s <signingcert>
```

- `signaturefile` specifies the path to the AIR application's `signatures.xml` file, located in the applications META-INF directory
- `<signingcert>` specifies the certificate used to sign the AIR application

## License Generator

Using the License Generator command line tool, you can generate licenses without requiring the client to send a license request to a server. A pre-generated license can be embedded into the content or delivered to the client through other mechanisms, such as a simple HTTP web server.

*Note: The License Generator is only included with Flash Access Professional.*

## Configuration File Properties

Before you run the License Generator, specify values for the License Generator properties. The configuration file specifies the following properties. For property names that include *n*, *n* represents an integer starting with 1 and increasing for each instance of the property.

Property	Description
<code>licensegen.sign.certfile</code>	The PKCS12 file containing the License Server credentials for signing licenses. This property should refer to a <code>.pfx</code> file containing a certificate and private key.
<code>licensegen.sign.certpass</code>	The password used to protect the file specified by <code>licensegen.sign.certfile</code> .
<code>licensegen.domainca.n</code>	If generating domain-bound licenses, one or more Domain CA certificates must be specified to indicate the domain authorities trusted by this license issuer. If the license recipient is a domain certificate, which was not issued by one of the specified Domain CAs, a license cannot be generated. This property specifies a <code>.cer</code> file that contains the certificate only (either PEM or DER format is acceptable). <i>n</i> must be monotonically increasing, starting from 1.
<code>licensegen.keys.asymmetric.licenseServerCredential.n</code>	Optional PKCS12 file containing additional License Server credentials for decrypting the CEK in the metadata and policy. Additional credentials may be configured if content was previously packaged with a License Server certificate other than that specified by <code>licensegen.sign.certfile</code> . This property should refer to a <code>.pfx</code> file containing a certificate and private key. <i>n</i> must be monotonically increasing, starting from 1.
<code>licensegen.keys.asymmetric.licenseServerCredential.n.password</code>	The password used to protect the file specified by <code>licensegen.keys.asymmetric.licenseServerCredential.n</code> .

## Command line usage

To generate a license, use the following syntax:

```
java -jar AdobeLicenseGenerator.jar -m metadata[options]
```

`metadata` is a `.metadata` file containing the Flash Access DRM metadata. This file can be obtained from protected content using the `-d -m` option of Media Packager.

To display a previously generated license, use the following syntax:

```
java -jar AdobeLicenseGenerator.jar -d license
```

`license` is a file containing a Flash Access license generated by the License Generator.

The following table describes the command line options that can be specified along with the syntax previously mentioned:

Command Line Option	Description
<code>-c configfile</code>	Specify the location of the configuration file. If this option is not used, the License Generator will look for <code>flashaccesstools.properties</code> in the working directory. Options specified on the command line take precedence over those present in the configuration file.
<code>-d licensefile</code>	Show information about a license that was already generated.
<code>-leaf leaf-filename</code>	Generate a leaf license and write the output to a specified file.
<code>-m metadata-filename</code>	Specify the content metadata for which to generate a license. (Required to generate license)
<code>-noprompt</code>	Do not ask if the destination file should be overwritten. If the destination file already exists and <code>-o</code> is not set, an error will be returned.
<code>-o</code>	If the destination file already exists, overwrite it without prompting.
<code>-policy policy-num</code>	If the metadata contains multiple policies, specify the number of the policy to use (starting at 1) to generate the license. If not specified, the first policy is used.
<code>-r recipient-cert</code>	Generate a license for the specified recipient. A device or domain certificate may be used. Multiple <code>-r</code> options may be specified to create a license for multiple recipients.
<code>-root root-filename</code>	Generate a root license and write the output to the specified file.

## License Embedder

Using the License Embedder command line tool, you can embed pre-generated licenses into content protected using Media Packager.

**Note:** *The License Embedder is only included with Flash Access Professional.*

### Command line usage

To embed a license, use the following syntax:

```
java -jar AdobeLicenseEmbedder.jar sourcefile destfile [options]
```

- `sourcefile` is an encrypted FLV or F4V file.
- `destfile` specifies where the encrypted content with the embedded license will be written. If a directory is specified, the file will be saved in this directory using the same filename as the source file, but the directory must not be the directory which contains the source file.

The following table describes the command line options that can be specified along with the syntax previously mentioned:

Command Line Option	Description
<code>-l license-filename</code>	Name of the file containing license to embed. Multiple <code>-l</code> options may be specified to embed multiple licenses.
<code>-m metadata-filename</code>	Specify the content metadata for which to generate a license. (Required to generate license)
<code>-noprompt</code>	Do not ask if the destination file should be overwritten. If the destination file already exists and <code>-o</code> is not set, an error will be returned.
<code>-o</code>	If the destination file already exists, overwrite it without prompting.

## License server and watched folder packager

The reference implementation server can help you create a license server using the Flash Access SDK. In this implementation, users are authenticated based on user entries in a database. The server includes demonstration business logic for issuing licenses. It also implements compatibility support for Flash Media Rights Management Server 1.0 and 1.5.

The reference implementation server also includes a watched folder implementation of the packager. This component may be deployed along with the license server or on a separate machine. With this packager implementation, multiple watched folders can be created. When content is dropped into the watched folder, the packager automatically packages the content.

The license server and packager are deployed as separate WAR files, so you can choose whether to run them on separate servers or in a single Apache Tomcat® instance. The license server is in the `flashaccess.war` and the packager is in `flashaccess-packager.war`. The optional `edcws.war` contains support for license requests from FMRMS 1.x clients.

The Reference Implementation sample code demonstrates the following features:

- License Server:
  - Handling authentication requests, using a database to validate username/password
  - Handling license requests and determining which type of license to issue when license chaining is used.
  - Issuing licenses for content containing multiple policies
  - Using database to determine if user is authorized to view content
  - Using policy update lists
  - Using machine revocation lists
  - Using an HSM or PKCS12 file to store credentials
  - Encrypting passwords specified in properties file
  - Specifying multiple license server or transport credentials (after credentials are renewed, the old credentials are kept on the server so existing content can be consumed without needing to repackage)
  - Restricting DRM/Runtime versions allowed to make requests to the license server
  - Setting client clock windback preferences
  - Restricting time difference allowed between request time and server time (to prevent replay attacks)
  - Handling requests from 1.x clients (triggers 1.x client to upgrade to 2.0)
  - Converting 1.x metadata to 2.0 metadata on the fly, using 1.x license information stored in a database

- Sample code for converting 1.x policies to 2.0 policies
- Sample scripts for importing 1.x license information from an existing database
- Get Server Version
- Domain registration
- Domain de-registration
- Synchronization requests
- Packager Server:
  - Implementing a packager implementation that automatically packages content added to a watched folder
  - Using an HSM or PKCS12 file to store credentials
  - Encrypting passwords specified in properties file
  - Configuring the packager, creating policies, and creating policy update lists using an AIR application

## Requirements

You will need to ensure that you have the following installed:

- Tomcat 6.0 or later
- A database such as MySQL (only required for License Server component)
- Java 1.6
- Ant (to use the sample build scripts)

Once you have installed Tomcat and MySQL, obtain the credentials from Adobe.

## Building the license server

The reference implementation license server includes WAR files for deploying the license server. It also includes all the license server source code and an Ant build script (Reference Implementation\Server\refimpl\build-refimpl.xml) so you can easily make changes to the code.

*Note: This step is only needed if you want to modify the source code. For evaluation purposes, you can skip this step and use the WAR files as shipped.*

Before running the Ant script, modify the script to specify the locations of the Flash Access SDK, Tomcat, MySQL, and Log4J. Open build-refimpl.xml in a text editor and edit the values of the properties sdkdir, tomcatdir, mysqldir, and log4jdir. To compile the source code and create the WAR files for the reference implementation, run the script using "ant -f build-refimpl.xml all" in the directory containing the Ant script. When the script is complete, a refimpl-build/wars directory containing the server WAR files will be created.

## Verifying the license server

To verify the license server installation, open the URL <http://yourip:8080/flashaccess/license/v3> from your Internet browser. You will get a "License Server is setup correctly" message.

## Set up a domain server

To configure the domain server on an existing license server installation, perform the following tasks:

- 1 Open the *flashaccess-reimpl.properties* file under *tomcat/lib*.

- 2 Under the 'Domain CA certificate' option, fill the Domain CA certificate details. This certificate will be used for accepting the domain tokens.
- 3 Under the 'Domain CA credential' option, fill the Domain CA credential certificate (PFX) details. This certificate will be used for signing domain certificates and tokens.
- 4 Specify the value for *DomainServerURL*. If this value is NULL, the domain authentication will succeed. However, while joining the domain, there will be a join domain error.

## Configuration

You will need to configure the server properties files, watched folder properties, set up the database, and configure the HSM.

### Server properties files

The server requires two configuration files, one for the license server and one for the packager. Both files must be placed on the classpath. The properties files contain the location of the credentials issued by Adobe. These credentials can be specified as a .pfx file and password or by providing an alias and password for a credential stored on an HSM.

Please refer to the property files for details about the specific values and usage of the each parameter. Sample properties files can be found in the "resources" directory of the reference implementation (Reference Implementation\Server\resources).

To ensure the security of your credential's password, a tool is provided (ScrambleUtil.class) to encrypt the password before it is entered into the flashaccess-refimpl.properties or flashaccess-refimpl-packager.properties file.

To properly prepare your credential's password:

- 1 Go to Reference Implementation\Server\refimpl\scrambler.
- 2 From the command prompt, enter the command:

```
java -classpath path_to_adobe-flashaccess-sdk.jar;.
com.adobe.flashaccess.refimpl.util.ScrambleUtil "your_pfx_password"
```

**Note:** The previous example uses a semicolon (;) as the delimiter. For platforms other than Microsoft Windows, use a colon (:) as the delimiter.

The utility outputs the encrypted password, which you must copy to the .properties file.

### Preparing passwords for the Server properties files

To ensure the security of your credential's password, a tool is provided to encrypt the password before it is entered into the flashaccess-refimpl.properties or flashaccess-refimpl-packager.properties file.

To run the tool using the ANT script provided:

- Go to <Reference Implementation Server Path>\refimpl
- Ensure the "sdkdir" property in build-refimpl.xml points to the directory containing the Flash Access 3.0 SDK
- Run the following command using ANT 1.8 or higher:

```
ant -f build-refimpl.xml
```

- When prompted, type your credential's password

To run the tool using Java:

- Go to <Reference Implementation Server Path>\scrambler

- From the command prompt, enter the command:

- On Windows:

```
java -classpath path_to_adobe-flashaccess-sdk.jar; .
com.adobe.flashaccess.refimpl.util.ScrumbleUtil your_pfx_password
```

- On Linux:

```
java -classpath path_to_adobe-flashaccess-sdk.jar: .
com.adobe.flashaccess.refimpl.util.ScrumbleUtil your_pfx_password
```

The utility outputs the encrypted password, which you must copy to the .properties file.

**Note:** Passwords encoded using the password scrambling utility provided with the reference implementation will not work with the Flash Access Server for Protected Streaming.

### License server properties file

The flashaccess-refimpl.properties file is used to configure the License Server component of the reference implementation. At a minimum, be sure to configure the properties related to the Transport Credential and the License Server Credential. The locations of the credential files must be specified relative to the directory specified by the "config.resourcesDirectory" property. This file also contains several properties related to packaging content: these properties are only used for Flash Media Rights Management Server 1.x metadata conversion. If you modify any of the values in this property file, you need to restart the license server for the changes to take effect.

The following properties have been added/modified in Flash Access 3.0 (not the complete list):

Property	Description
HandlerConfiguration.minSupportedClientVersion	(Optional) Specify the minimum client version allowed to talk to the server and use licenses issued by the server.
HandlerConfiguration.DRMModuleRequirements.ExcludedVersions.n	Supports OS, release, arch, model, vendor, env, screen.
HandlerConfiguration.RuntimeModuleRequirements.ExcludeVersions.n	Supports application, OS, release, arch, model, vendor,, env, screen.
HandlerConfiguration.DomainCAs.n	Specify Domain CAs from which the license server will accept domain tokens. Required in order to issue domain-bound licenses.
DomainRegistrationHandler.ServerCredential	The PKCS12 file containing Domain CA credentials for signing domain tokens. This property should refer to a .pfx file containing a certificate and private key. Required to process domain registration request.
DomainRegistrationHandler.ServerCredential.password	The password used to protect the file specified by DomainRegistrationHandler.ServerCredential.
DomainRegistrationHandler.DomainServerUrl	Domain server base URL (protocol/host/port). Required to process domain registration requests.
RefImpl.HSM.HandlerConfiguration.DomainCAs.Alias.n	Aliases of authorized Domain CA certificates stored on HSM. When HSM is enabled, use this property instead of HandlerConfiguration.DomainCAs.n
RefImpl.HSM.DomainRegistrationHandler.ServerCredential.Alias	Alias of Adobe-issued domain CA credential (certificate and private key) stored on HSM. When HSM is enabled, use this property instead of DomainRegistrationHandler.ServerCredential and DomainRegistrationHandler.ServerCredential.password

### Packager properties file

The flashaccess-refimpl-packager.properties is used to configure the Watched Folder Packager component of the reference implementation. At a minimum, be sure to set the license server URL, license server certificate, packager credential, and key protection options. This file also contains the location of each watched folder (packager.watchfolder.source.*n*). Any changes made to the values in this property file will take effect the next time the watched folder packager runs (restarting the server is not required). However, if there is a configuration error in the packager, the watched folder packager thread will exit, and the server will need to be restarted to restart the packager thread.

### Watched folder properties

Each watched folder contains a file called "properties/watchedfolder.properties". This file contains the packaging options for content placed in this folder, including what to encrypt and which policies to apply. Any changes made to the values in the property file take effect the next time the watched folder packager runs (you do not need to restart the server).

If there is a configuration error in the packager properties file, the packager thread stops. To resume the watched folder packager, restart the server. If there is a configuration error in a watched folder properties file, the watched folder is temporarily removed from the list of folders the packager processes. To add the watched folder back to the list, restart the server or modify the packager properties file. If an error occurs during packaging of a particular file (for example, because the file is corrupt), the file is skipped and the remaining files in the folder are processed.

### Setting up the database and configuring the JNDI datasource

The reference implementation license server requires a database to support the following features:

- User authentication
- Usage model demo business rules
- Metadata conversion
- Domain support

Anonymous license acquisition does not require a database to be running.

*Note: The instructions in this section are for the Microsoft Windows platform. For other operating systems, see the documentation for your operating system or see the MySQL documentation.*

To run the license server, you will need to install and configure MySQL 5.1.34:

- 1 Run the MySQL installer (found in the Third Party\MySQL\Installer\5.1 folder on the DVD).
- 2 At the end of the installation procedure, check "Configure MySQL Server Now" to start the configuration wizard. Use the default settings or select specific settings for your testing purposes, with the exception that on the 5th screen you must select "Online Transaction Processing (OLTP)" or "Manual Setting" and enter the maximum number of connections allowed.
- 3 Make a note of the root password.
- 4 If you need to re-install MySQL, follow these steps to avoid problems in starting the server afterward:
  - Delete the folder system drive \Documents and Settings\All Users\Application Data\MySQL.
  - Delete the old MySQL install folder: for example, system drive:\Program Files\MySQL\MySQL\Server 5.1.

Next, you will need to install MySQL JDBC Driver 5.1.7. To do this, copy mysql-connector-java-5.1.7-bin.jar (found in the Third Party\MySQL\Installer\5.1 folder on the DVD) to Tomcat Server lib directory: ...\\Tomcat6.0\lib.

*Note: MySQL JDBC Driver 5.1.7 works with Tomcat 6.0. Older versions of Tomcat are not supported.*

Set up the sample database by setting up the database schema and populating the database with sample data. To do this, perform the following steps:

- 1 Go to Window's Start Menu, MySQL -> MySQL Server 5.1 -> MySQL Command Line Client.
- 2 After typing in the password, execute the following SQL script to add the user account `dbuser` for establishing a connection through a web application and create database schema (make sure that there is no ";" at the end. Just press enter.):

```
mysql> source Reference Implementation\Server\dbscript\createsampledbsql
```

- 3 Edit the script that populates sample data in the tables to include data for your testing purposes: Reference Implementation\Server\dbscript\PopulateSampleDB.sql.
- 4 Execute this script to populate the data as you did in step 2.

**Note:** The first time you run the `CreateSampleDB.sql` script you will receive the following error:

```
ERROR 1396 (HY000): Operation DROP USER failed for 'dbuser'@'localhost' Query OK, 0 rows affected (0.00 sec).
```

You can safely ignore this error. This only happens the first time you run this script.

At this point you will need to configure Database Connection Pooling (DBCP). DBCP uses the Jakarta-Commons Database Connection Pool. A JNDI Datasource TestDB is configured to take advantage of this application server connection pooling. To change database connection to point to a MySQL server that is not on localhost, modify the `META-INF/context.xml` file (which specifies the location, username, and password of the license server's database) located in `flashaccess.war`, or modify `\Reference Implementation\Server\refimpl\WebContent\META-INF/context.xml` and recreate the WAR file using the updated files. To change any of these parameters, edit the `context.xml` located in the `WebContent` directory and use the Ant script to recreate the WAR file. To tune the database, change the JNDI datasource settings in this file.

If you debug the Reference Implementation project within Eclipse, you need to add `$CATALINA_HOME\lib\tomcat-dbc.jar` to your run/debug configuration. This step is not required if you run the `flashaccess.war` file on a standalone Tomcat 6.0 server.

## HSM configuration

Use of an HSM is not required, but it is recommended. The reference implementation can be configured to use the Sun PKCS11 provider for HSM support. In order to use a credential on an HSM, you must create a configuration file for the Sun PKCS11 provider. See the Sun documentation for details. To verify that your HSM and Sun PKCS11 configuration file are configured properly, you can use the following command (keytool is installed with the Java JDK):

```
keytool -keystore NONE -storetype PKCS11 -providerClass sun.security.pkcs11.SunPKCS11 -  
providerArg pkcs11.cfg -list
```

If you see your credentials in the list, the HSM is configured properly.

**Note:** If you use a 64-bit version of Windows, HSM is currently not supported by the Reference Implementation.

## Crossdomain policy file

In order for Flash Runtime clients to request a license from the License Server, a crossdomain policy file is required. See *Using Flash Access Server for Protecting Content* for more details.

## Deploying the license server and watched folder packager

Copy the license server WAR files to Tomcat's webapps directory. If you have previously deployed the WAR file, you may need to manually delete the unpacked WAR directories ("flashaccess", "edcws", and "flashaccess-packager" in Tomcat's webapps directory). To prevent Tomcat from unpacking WAR files, edit the server.xml file in Tomcat's conf directory and set the "unpackWARs" attribute to "false".

The properties file (flashaccess-refimpl.properties) must be on the classpath for the server to load the properties. Copy this file to a directory and update the file with the appropriate values. Edit the catalina.properties file in Tomcat's conf directory and add the directory containing flashaccess-refimpl.properties to the "shared.loader" property. The log4j.xml file for configuring logging must also be on the classpath (see resources\log4j.xml for an example).

The reference implementation server uses several certificate files, policy files, and other resources. Those files are all located in one resource folder. By default, the resource folder is C:\flashaccess-server-resources, but this location can be modified in flashaccess-refimpl.properties. Be sure to copy all the required resources to this location before starting the server.

To start Tomcat and the license server, run "catalina.bat start" from Tomcat's bin directory.

## Troubleshooting

Listed below are common problems and solutions for deployment:

- If you see the following error:

```
"Error decoding the password for
HandlerConfiguration.ServerTransportCredential.password
    javax.crypto.IllegalBlockSizeException: Input length must be multiple of 8 when
decrypting with padded cipher"
```

Make sure the password is encrypted using the provided ScrambleUtil class.

- If you see the following error:

```
"Unable to load credential from file.pfx -- possibly wrong password."
```

Make sure you specified the correct encrypted password for the PFX file.

- If you see the following error:

```
"javax.crypto.BadPaddingException: Given final block not properly padded"
```

Make sure you used the password scrambler class provided with the Reference Implementation (this scrambler utility is different from the one provided with the Adobe® Flash® Access™ Server for Protected Streaming).

## Determining if Reference Implementation License Server is properly running

There are several ways to determine whether your server has started correctly. Viewing the catalina.log logs may not be sufficient, as the license server logs to its own log files. Follow the steps below to ensure your Reference Implementation has started up properly.

- Check your "AdobeFlashAccess.log" file. This is where the Reference Implementation writes log information. The location of this log file is indicated by your log4j.xml file and can be modified to point to any location you'd like. By default, the log file will be output to the working directory where you've run catalina.
- Navigate to the following URL: <http://your server:server port/flashaccess/license/v1>. You should see the text "License Server is setup correctly".

Another way to test if your server is running correctly is to package a piece of test content, set up a sample video player, and play it. The following procedure describes this process:

1 Navigate to \Reference Implementation\Command Line folder. For information on installing the command line tools, see "Installing the command line tools".

2 Create a simple anonymous policy by using the following command:

```
java -jar libs\AdobePolicyManager.jar new policy_test.pol
```

For mor information on creating policies using the Policy Manager, see "Command line usage".

3 Set the `encrypt.license.serverurl` property in the `flashaccesstools.properties` file to the URL of the license server (for example, `http://localhost:8080/`). The `flashaccesstools.properties` file is located under the \Reference Implementation\Command Line Tools folder.

4 Package a piece of content by using the following command:

```
java -jar libs\AdobePackager.jar test_input_FLV output_file
```

5 Copy the 2 generated files to the Tomcat webapps\ROOT\content folder.

6 Extract SampleVideoPlayers.zip and copy FlashPlayer\Release to the Tomcat webapps\ROOT\SVP\ folder.

7 Install Flash Player 10.1 or later.

8 Open the web browser and navigate to the following URL:

```
http://localhost:8080/SVP/SampleVideoPlayer_FP.html
```

9 Navigate to the following URL, then click the Play button:

```
http://localhost:8080/Content/your_encrypted_FLV.
```

10 If the video fails to play, check if any error codes were written in the logging pane of the Sample Video Player, or in the `AdobeFlashAccess.log` file. The location of the `AdobeFlashAccess.log` log file is indicated by your `log4j.xml` file, and can be modified to point to any location you'd like. By default, the log file is written to the working directory where you've run catalina.

## Implementing the usage models

The Reference Implementation includes business logic for demonstrating how to enable the following four different usage models for a piece of packaged content:

- Download-to-own (DTO)
- Rental/Video-on-demand (VOD)
- Subscription (all-you-can-eat)
- Ad-funded

To enable the usage model demo, specify the custom property "`RI_UsageModelDemo=true`" at packaging time. If you are packaging content using the Media Packager command line tool, specify:

```
java -jar AdobeMediaPackager.jar source.flv dest.flv -k RI_UsageModelDemo=true
```

**Note:** If you do not activate the optional demo mode at packaging time, the license server uses the policy specified at packaging time to issue a license. If multiple policies were specified, the license server uses the first valid policy.

In the demo, the business logic on the server controls the actual attributes of the licenses generated. At packaging time, only minimal policy information must be included in the content. Specifically, the policy only needs to indicate whether authentication is required to access the content. To enable all four usage models, include one policy that allows anonymous access (for the Ad-funded model) and one policy that requires user name/password authentication (for the other 3 usage models). When requesting a license, a client application can determine whether to prompt the user for authentication based on the authentication information in the policies.

To control the usage model under which a particular user is to be issued a license, entries may be added to the Reference Implementation database. The Customer table contains user names and passwords for authenticating users. It also indicates whether the user has a subscription. Users with subscriptions will be issued licenses under the Subscription usage model. To grant a user access under the Download to Own or Video on Demand usage models, an entry may be added to the CustomerAuthorization table, which specifies each piece of content the user is allowed to access and the usage model. See the PopulateSampleDB.sql script for details on populating each table.

When a user requests a license, the Reference Implementation server checks the metadata sent by the client to determine if the content was packaged using the `RI_UsageModelDemo` property. If so, the following business rules are used:

- If one of the policies requires authentication:
  - If the request contains a valid authentication token, look for the user in the Customer database table. If the user was found:
    - If the `Customer.IsSubscriber` property is `true`, generate a license for the Subscription usage model and send it to the user.
    - Look for a record in the CustomerAuthorization database table for this user and content ID. If a record was found:
      - If `CustomerAuthorization.UsageType` is `DTO`, generate a license for the Download To Own usage model and send it to the user.
      - If `CustomerAuthorization.UsageType` is `VOD`, generate a license for the Video On Demand usage model and send it to the user.
  - If none of the policies allow anonymous access:
    - If there is not a valid authentication token in the request, return an "authentication required" error.
    - Otherwise return a "not authorized" error.
- If one of the policies allows anonymous access, generate a license for the Ad-funded usage model and send it to the user.

Before the Reference Implementation server can issue licenses for the usage model demo, the server needs to be configured to specify how licenses are generated for each of the four usage models. This is done by specifying a policy for each usage model. The Reference Implementation includes four sample policies (`dto-policy.pol`, `vod-policy.pol`, `sub-policy.pol`, `ad-policy.pol`) or you may substitute your own policies. In `flashaccess-refimpl.properties`, set the following properties to specify the policy to use for each usage model and place the policy files in the directory specified by the `config.resourcesDirectory` property:

```
# Policy file name for Download To Own usage
RefImpl.UsageModelDemo.Policy.DTO=dto-policy.pol
# Policy file name for Rental usage
RefImpl.UsageModelDemo.Policy.VOD=vod-policy.pol
# Policy file name for Subscription usage
RefImpl.UsageModelDemo.Policy.Subscribe=sub-policy.pol
# Policy file name for Ad Supported (free) usage
RefImpl.UsageModelDemo.Policy.Free=ad-policy.pol
```

## Download-to-own

With the DTO usage model, a user may download the content for use online or offline and is issued a permanent license for the content. When requesting a license, the user must authenticate so the server can verify that the user has purchased the content.

## Rental/Video-on-demand

With the VOD usage model content is offered with time-based restrictions. For example, a user has the option to play the content during a 30-day period; however, once playback begins, the user has up to 48 hours to finish watching, after which time the content will no longer be playable. When requesting a license, the user must authenticate so the server can verify that the user has a rental account.

## Subscription

Some services offer paid subscriptions that give users unlimited access to a large library of content for as long as they continue to pay the monthly fees. The license server issues a unique license for each piece of content and also issues a root license whose expiration coincides with the subscription period. Each month, when the user renews his subscription, the root license can also be renewed. When requesting a license, the user needs to authenticate so the server can verify that the user's subscription is up to date.

## Ad-funded

Content is monetized by including advertising as part of the experience. With this model, content can be distributed without requiring user authentication.

## Implementing domain registration

The reference implementation license server demonstrates two variations on domain registration business logic. A typical domain server would implement only one of these workflows:

- Identity-based domains
- Anonymous domains

### Identity-based domains

In this use case, each authenticated user has his own domain, and a certain number of devices are allowed to join the domain. To use this type of domain with the reference implementation, create a policy specifying domain registration is required. Specify your server's host and port for the domain server URL and specify username/password authentication is required.

The reference implementation implements the following logic for domain registration:

- 1 Determine the domain name to assign to this user. The domain name will be *namequalifier:username* extracted from the authentication token. If there is no authentication token, return error DOM\_AUTHENTICATION\_REQUIRED (503).
- 2 Lookup the domain name in the `DomainServerInfo` table. If an entry is not found, insert an entry into the table (default values are authentication required, max domain membership=5).
- 3 Check if the device has already registered with the domain:
  - a Lookup the domainname in the `UserDomainMembership` table. For each machine ID found, compare with the machine ID in the request. If this is a new machine, add an entry to the `UserDomainMembership` table. Next, find the matching records in `UserDomainRefCount` table. If an entry does not exist for this machine GUID, add a record.



- 5 Lookup all the domain keys for this domain in the `DomainKeys` table.
  - a If `DomainServerInfo` indicates the keys need to be rolled over, generate a new key pair, save it in the `DomainKeys` table (with key version one higher than the highest existing key), and reset the “Key Rollover Required” flag in `DomainServerInfo`.
  - b For each domain key, generate a domain credential.

The reference implementation implements the following logic for domain de-registration:

- 1 Parse the domain name from the request URL.
- 2 Lookup the requested domain name in the `DomainServerInfo` table.
- 3 If authentication is required for the requested domain, make sure a valid authentication token was included in the request, and match the Auth Namespace, if specified in the database.
- 4 Lookup the domain name and machine GUID in the `DomainMembership` table. If a matching entry is not found, return error `DEREG_DENIED` (401).
- 5 If this is not a preview request, delete the entry from `DomainMembership` and set the “Key Rollover Required” flag in `DomainServerInfo`.

In this use case, since a large number of machines could join the domain, completely matching the machine ID is not possible. Instead, the random machine GUID assigned to the machine during individualization is used.

## Migrating from FMRMS 1.0 or 1.5 to Flash Access 3.0

In order to continue to issue licenses for content packaged using Flash Media Rights Management Server (FMRMS) 1.0 or 1.5, license and policy data must be migrated from the LiveCycle ES server to the customer's new server based on the Flash Access 3.0 SDK. The important steps are:

- 1 Importing license information
- 2 Converting FMRMS policies to Flash Access 3.0 format
- 3 Supporting the 1.x compatibility requests via the `FMRMSv1RequestHandler` and `FMRMSv1MetadataHandler`

To import license information from LiveCycle ES into your Flash Access 3.0-based server, refer to the sample database scripts provided in the Reference Implementation\Server\migration\db folder. Sample scripts are provided for exporting the relevant data from a MySQL, Oracle, or SQL Server database into a CSV file format. Once the data is exported, it can be imported into the database of your choice. The exported license information includes the License ID, a Content ID assigned at packaging time, the ID of the Policy used, the time the content was packaged, and the content encryption key. For 3.0, this information is required in order to convert the 1.x content metadata into the 3.0 metadata format (see `FMRMSv1RequestHandler` and `FMRMSv1MetadataHandler`). In the reference implementation, this data is stored in the License database table and used by `RefImplMetadataConvReqHandler`.

Existing policies will need to be converted to the Flash Access 3.0 format in order to use those policies when converting metadata and issuing licenses for 1.0 or 1.5 content. The Reference Implementation\Server\migration folder contains sample code for creating a 3.0 policy based on older policies.

If you are migrating from FMRMS 1.0 to Flash Access 3.0, see the `V1_0PolicyConverter.java` sample. Compile the sample code by running `ant -f build-migration.xml build-1.0-converter` (the script expects the 1.0 and 3.0 libraries to be in `libs/1.0` and `libs/3.0` respectively). Edit the `converter.properties` file to point to your LiveCycle ES server. Then run `ant -f build-migration.xml migrate-all-1.0-policies` to convert all FMRMS 1.0 policies to 3.0 format.

If you are migrating from FMRMS 1.5 to Flash Access 3.0, see the `V1_5PolicyConverter.java` sample. Compile the sample code by running `ant -f build-migration.xml build-1.5-converter` (the script expects the 1.5 and 3.0 libraries to be in `libs/1.5` and `libs/3.0` respectively). Edit the `converter.properties` file to point to your LiveCycle ES server. Then run `ant -f build-migration.xml migrate-all-1.5-policies` to convert all FMRMS 1.5 policies to 3.0 format.

The converted policies will be written to a set of files. In addition, `PolicyConverter` will output a CSV file containing the mapping of old policy IDs to new policy IDs. This file can be imported into the "PolicyConversion" table in the reference implementation database, and will be used by `RefImplMetadataConvReqHandler`.

Once the relevant data has been migrated to your Flash Access 3.0-based server, you are ready to implement support for 1.x compatibility requests. See `RefImplUpgradeV1ClientHandler` and `RefImplMetadataConvReqHandler` in the reference implementation for examples of how to process these types of requests.

## Upgrading existing deployments

To upgrade a server running the version 2.0 Reference Implementation License Server or Watched Folder Packager, replace the `.war` files deployed on your Application Server with the files included with Flash Access 3.0 Reference Implementation Server.

If you plan to use domain registration with the Reference Implementation License Server, several new database tables are required. To re-create the entire reference implementation database, run `CreateSampleDB.sql`. To preserve the existing database records and add the new tables, open `CreateSampleDB.sql` and only run the commands to create the following tables:

- `DomainServerInfo`
- `DomainKeys`
- `DomainMembership`
- `UserDomainMembership`
- `UserDomainRefCount`

The following properties must be added to `flashaccess-refimpl.properties` to use the domain support:

- `HandlerConfiguration.DomainCAs.n` OR `RefImpl.HSM.HandlerConfiguration.DomainCAs.Alias.n`
- `DomainRegistrationHandler.ServerCredential` and `DomainRegistrationHandler.ServerCredential.password`, OR `RefImpl.HSM.DomainRegistrationHandler.ServerCredential.Alias`
- `DomainRegistrationHandler.DomainServerUrl`

## Adobe Flash Access Manager AIR application usage

The Flash Access Manager is an Adobe AIR application for packaging Flash Access content. Using this application, you can create policies, manage a policy update list, and package content. You can also set up Watched Folders to automatically package content with certain settings when new content is added to the folder.

## Building the Packager Server and AIR Application

There are two components required to use the Adobe Flash Access Manager: the Adobe Flash Access Manager AIR application and the Packager Server (flashaccess-packager.war). Both components are distributed in both source and binary forms with the Reference Implementation.

### Building the Packager Server

If you wish to modify the source code, see the instructions on compiling the Reference Implementation in "[Building the license server](#)" on page 16".

### Building the Flash Access Manager AIR Application

To build the Flash Access Manager AIR file from the source code, you need the Flex and AIR SDK installed on your machine. Before you can package and run the application, you must compile the MXML code into a SWF file using the amxmlc compiler. The amxmlc compiler can be found in the bin directory of the Flex 4 or later SDK. If desired, you can set your path environment variable to include the Flex SDK bin directory to make it easier to run the utilities on the command line.

Use the following procedure to build the Flash Access Manager AIR file:

- 1 Open a command shell or a terminal and navigate to the project folder of the Flash Access Manager AIR application (UI Tools\Flash Access Manager in the Reference Implementation directory).
- 2 Enter the following command:

```
amxmlc src\FlashAccessmanager.xml
```

Running amxmlc produces FlashAccessManager.swf, which contains the compiled code of the application.

The Adobe AIR SDK includes the AIR Developer Tool (ADT) utility to package AIR applications and generate certificates. AIR applications should be digitally signed; users will receive a warning when installing applications that are not properly signed or are not signed at all. To generate a certificate using the command line, open a console window in the same folder as your AIR application and type the following:

```
adt -certificate -cn SelfSigned 1024-RSA testCert.pfx some_password
```

Substitute *some\_password* with a password of your choice. After a few seconds, ADT should complete its certificate generation process and you should have a new testCert.pfx file in your application directory.

Next, use ADT to package the application into an .air file, by using the command:

```
adt -package -storetype pkcs12 -keystore testCert.pfx FlashAccessManager.air  
src\FlashAccessManager-app.xml . -C src assets
```

This command tells ADT to package your application, using the key file in testCert.pfx. In the line above, you configure ADT to package your entire application into a file named FlashAccessManager.air, and to include the files FlashAccessManager-app.xml and FlashAccessManager.swf and the images from the assets directory.

As part of this process, you'll be prompted for the password that you set for your new certificate file. Enter it, wait a moment, and a FlashAccessManager.air file should appear in the same directory as your project files.

## Initial Flash Access Manager setup

Use the following procedure to set up Flash Access Manager:

- 1 Deploy the Packager Server. This server should only be available to users within your firewall (do not deploy this software on a public-facing machine). For more information on deploying the server, see [“Deploying the license server and watched folder packager”](#) on page 21".

- Copy flashaccess-packager.war to Tomcat's webapps folder.
- Copy flashaccess-refimpl-packager.properties from resources to a location on the classpath.
- Start the server. You will see some errors due to problems in the properties file; this is expected since the properties have not been filled in yet.

- 2 Install the Adobe Flash Access Manager AIR application by launching the .air file (requires AIR 1.5 or higher).
- 3 Launch the Adobe Flash Access Manager AIR application.

If your server is running somewhere other than `http://localhost:8080`, you see errors stating that the application cannot connect to the server. Dismiss the error dialog and fill in the correct URL for the "Packager Server URL" in the Preferences Tab. If the server is running at the specified URL and the properties file is on the classpath, the Preferences screen will be populated with the values in the properties file. After you set the packager server URL, the AIR application remembers this setting, and you will not have to enter it the next time you launch the application.

- 4 Fill in the values in the Preferences tab and click Save. For an explanation of each parameter, see [“Setting preferences”](#) on page 29".
- 5 If you want to use the Watched Folders, you will need to restart the server to recover from the errors you saw in step 3. If the preferences are configured properly, no errors should appear during startup.

## Setting preferences

With the exception of the Packager Server URL, all the preferences specified below are stored in the `flashaccess-refimpl-packager.properties` file on the server. All the settings can be modified either directly in the properties file or through the AIR application. Passwords are encrypted when they are stored in the properties file on the server. Type the unencrypted password into the UI, and it will be encrypted before it is stored in the file.

*Note: All directories and paths refer to directories on the packager server, not on the client running the AIR application.*

Any changes made here take effect immediately once the preferences are saved. There is no need to restart the server unless the Packager Thread terminated due to configuration problems.

The preference descriptions use the following terms:

Preference	Description
Packager Server URL	Location of server running flashaccess-packager.war, for example, <code>http://localhost:8080</code>
Resource Directory	Directory containing policies, certificates, credentials, and any other resources required for the packager server
License Server URL	URL of the server from which the client should request a license, for example, <code>http://mylicenseserver.com:8080</code>

## Packager Preferences

This tab contains settings required for packaging content. The following table describes these preferences:

Preference		Description
License Server Transport Certificate		The server transport certificate, issued by Adobe. This certificate is used to secure communications between the client and license server. The file must be located in the Resource Directory.
Enable HSM		Specifies whether certificates and credentials are stored on an HSM. If so, preferences related to certificates and credentials will be disabled, and the properties on the HSM tab must be specified.
Key Encryption Options		Specifies how the Content Encryption Key is encrypted at packaging time
	License Server Certificate	The License Server Certificate, issued by Adobe. The file must be located in the Resource Directory. The CEK is encrypted with the public key of the license server. Only holders of the license server private key may decrypt the CEK.
Packager Credential		The packager credential, issued by Adobe. This file is used to sign the metadata during packaging.
	File Name	The PKCS#12 (.pfx) file containing certificate and private key. The file must be located in the Resource Directory.
	File Password	Password for .pfx file
Global Watched Folder Properties		Specifies settings common to all Watched Folders configured on this server.
	Check Interval in Milliseconds	Specifies how often Watched Folders should check for new content to package. The server iterates through all the configured Watched Folders, then sleeps for this amount of time.
	Output File Name Suffix	Specifies a file extension to add to output files. For example, if ".out" is specified and the input file is "video.flv", the output file would be "video.out.flv".
	Backup Input Files	Specifies whether a copy of the original content should be saved. If this option is not selected, the original file will be deleted after packaging has been completed successfully.
	Input Backup Subfolder Name	If the Backup Input Files option is selected, specifies a folder where input files will be saved. This option specifies a folder name relative to the Watched Folder input directory. If the folder does not exist, it will be created during packaging.
	Overwrite Existing Output Files	Specifies whether the output file may be overwritten if a file already exists with the same name. If this option is not selected and the output file already exists, processing of the input file will be skipped.

### Policy Update List Preferences

This tab contains settings required for creating Policy Update Lists. The following table describes the preferences:

Preference	Description
License Server Credential	The License Server credential, issued by Adobe. This credential is used to sign Policy Update Lists.
File Name	The PKCS#12 (.pfx) file containing certificate and private key. The file must be located in the Resource Directory.
File Password	The password for .pfx file

## HSM Preferences

Preferences in this tab only need to be specified if the "Enable HSM" checkbox is selected in the Packager tab. The following table describes these preferences:

Preference	Description
Sun PKCS#11 Config File Name	The full path to the Sun PKCS#11 provider's configuration file. See the Java PKCS#11 Reference Guide on Sun's website for details on the contents of this configuration file.
Partition Password	The password for the HSM partition specified in the PKCS#11 configuration file.
License Server Certificate Alias	Alias for Adobe-issued license server certificate stored on HSM. This certificate is used to encrypt the CEK during packaging. Specify this instead of "License Server Certificate" in the Packager tab.
License Server Transport Certificate Alias	Alias for Adobe-issued server transport certificate stored on HSM. This certificate is used to secure communications between the client and license server. Specify this instead of "License Server Transport Certificate" in the Packager tab.
Packager Credential Alias	Alias for Adobe-issued packager credential (certificate and private key) stored on HSM. This is used to sign the metadata during packaging. Specify this instead of "Packager Credential" in the Packager tab.
License Server Credential Alias	Alias for Adobe-issued license server credential (certificate and private key) stored on HSM. This credential is used to sign Policy Update Lists. Specify this instead of "License Server Credential" in the Policy Update List tab. (This alias will likely be the same as "License Server Certificate Alias.")

## Policy creation

Before any content can be packaged, one or more policies must be created. For an overview of the usage rules that may be specified in a policy, see "Usage rules".

### Create a new policy

To create a new policy, click **New** and enter a policy name. Fill in the desired policy attributes (all settings are optional). When done, click **Save**. The policy will be saved as `policyname.pol` in the Resource Directory.

### Basic Policy Options

The following table describes the Basic Policy preferences:

Preference		Description
Policy Duration		Specifies the validity period of content protected with this policy.
	Start at	Licenses cannot be used until this date/time.
	End at	Licenses cannot be used after this date/time.
	End after	Specifies the amount of time a license is valid (in minutes), starting from the time it is packaged.
License Caching		Specifies whether licenses may be cached by the client.
	Delete at	Licenses cannot be used after this date/time.
	Delete after	Specifies the amount of time a license is valid (in minutes), starting from the time it the license is issued by the license server.
	Cache Indefinitely	License may be cached on the client indefinitely.
	No License Caching	License may not be cached by the client. A new license must be obtained from the server each time the user plays the content.
Authentication		
	Anonymous	No authentication is required to view the content.
	Authenticated	Username/password authentication is required.
Enable License Chaining		Allows a license to be updated using a parent root license for batch updating of licenses. Once the leaf license expires, the server may issue the client a root license, which will renew all content protected with this policy.

## Play Rights

The following table describes the Play Rights preferences:

Preference		Description
Playback Window		The duration a license is valid (in minutes) after the first time the user plays the protected content.
Output Protection		Controls whether output to external rendering devices should be protected. Analog and digital outputs can be specified independently.
Restrictions		Blacklist of client versions not permitted to play content. All columns are optional.
	DRM	Specifies a list of DRM versions that are not permitted to play protected content.
	Runtime	Specifies a list of Runtime versions that are not permitted to play protected content.
Minimum Security Level		
	DRM	Minimum DRM security level required to play protected content.

Preference		Description
	Runtime	Minimum Runtime security level required to play protected content.
Allowed Applications		Whitelist of client applications permitted to play content. If not applications are specified, any SWF or AIR application is allowed.
	SWF	List of SWF URLs permitted to play protected content.
	AIR	List of AIR applications permitted to play protected content. Publisher ID is required, the remaining fields are optional.

Flash Access Manager supports policies containing multiple Play Rights. To create a policy with more than one Play Right, use the "Add additional Play Right" button, and fill in the desired attributes for each Play Right.

When consuming a license, the client uses the first Play Right for which it meets all the requirements. Multiple play rights may be used to specify different restrictions for different operating systems. For example, it is possible to specify one right with Output Protection required for Windows (by blacklisting DRM versions on Macintosh and Linux) and to specify a second right with Output Protection "Use if available" on other platforms (by blacklisting DRM versions on Windows).

### Custom Data

The following table describes the Custom Data preferences:

Preference	Description
Custom Policy Properties	Specify custom properties, which the license server may use when issuing licenses.
Custom License Properties	Specify custom properties, which will appear in the license issued to the client. Client applications will have access to these properties.

### Update an existing policy

To update an existing policy, choose the filename from the drop down list and click Open. Modify any desired policy attributes. All attributes can be modified except those related to Authentication and License Chaining.

When done, click Save. The policy file in the Resource Directory will be replaced with the updated version.

*Note: Even if the policy name is changed, the name of the file in the Resource Directory will not be modified.*

### Delete a policy

To delete an existing policy, choose the filename from the drop down list and click Delete.

### Policy update list

You can use Policy Update Lists to communicate policy changes to a License Server. If a policy is modified after it is used to package content, it is desirable to have the License Server aware of the most recent version of the policy, so that version can be used to issue a license.

To create a Policy Update List for the first time, click "Add policies" to view all available policies on the server. For any policies that have been updated since they were used to package content, select the "update" radio button.

If you no longer want to use a policy to issue any licenses and the policy was already used to package content, you may wish to revoke the policy. To do so, select the "revoke" radio button. When the desired policies have been selected, choose "Create Policy Update List". A file called PolicyUpdateList.dat will be saved in the Resources Directory.

To modify an existing Policy Update List, click "Add policies" to view all available policies on the server. Choose the additional policies to add or revoke. Existing entries in the Policy Update List can be changed in the upper section of the screen. Policies that are marked "updated" may be changed to "revoked", but once a policy is "revoked", it cannot be changed back to "updated".

When the desired changes have been made, choose "Create Policy Update List", and the PolicyUpdateList.dat file is regenerated. If a policy is already in the policy update list and was updated since the last time the list was generated, the most recent version of the policy will be used when the Policy Update List is generated again.

## Package media

Use the Package Media tab to package content. The Packager Properties section displays the Packager settings that were entered in the Preferences tab. To modify these settings, go to the Preferences tab, change the settings, and Save.

If you want to package a single FLV or F4V file, choose the "Select Single File" option and enter the full path to the source file and full path where the encrypted file should be saved.

If you want to package all files in a folder, choose the "Select Single Folder" option. Specify the folder containing the source files. Only files in the Input Folder matching the "Input Media File Selection" criteria will be packaged (files in subfolders are not packaged). Choose to encrypt .flv files, .f4v files, or enter a custom regular expression (for example ".\*" encrypts all files in the folder). The encrypted files will be saved in the specified output folder, using the same filename as the original file.

**Note:** File paths must refer to files available to the packaging server. If you are running the Flash Access Manager on a different machine than the packaging server, you must specify a path that is accessible by the server (either located on a network drive or on the server itself).

The following table describes the Package Media preferences:

Preference	Description
Policy File Name(s)	Select one or more policies from the drop-down list to apply to the content. To select multiple policies, hold down the CTRL key while selecting policies.
Seconds Unencrypted	Specifies the number of seconds of content to leave unencrypted at the beginning of the file. To encrypt starting from the beginning, enter "0".
Encrypt Video	Select this checkbox to encrypt video data
Encryption Level	If video encryption is enabled, select the encryption level for video data. High encrypts all video data. Medium and Low selectively encrypt portions of the video. (Only for F4V with H.264 video)
Encrypt Audio	Select this checkbox to encrypt audio data
Encrypt Script	Select this checkbox to encrypt script data (FLV only)
Custom Properties	Specify custom properties to include in the packaged content. These properties will be available to the license server when issuing a license. (Optional)

After the packaging options are selected, click the "Package Media" button to begin packaging the files.

## Watched Folders

You can use Watched Folders to automatically package content created in certain folders. Each Watched Folder can be configured with different packaging options. To test packaging options before creating a Watched Folder, use the Package Media tab.

To create a Watched Folder, click "Add New Watched Folder" and fill in the packaging options. See the "Packaging media files" section for a description of each option. When done, click "Save Watched Folder Properties".

When a Watched Folder is saved, the packaging options are saved to *[Input Folder]\properties\watchfolder.properties*. Any content added to the Input Folder which meets the Input Media File Selection criteria will automatically be packaged and placed in the Output Folder. See the Global Watched Folder Preferences in the section "Packager Preferences" to configure additional Watched Folder options.

To modify Watched Folder settings, select the Watched Folder input path from the list at the top of the screen. Modify the settings and click "Save Watched Folder Properties".

To delete a Watched Folder, select the Watched Folder input path from the list at the top of the screen and click "Delete Watched Folder Properties".