CA Federation Manager

Java SDK Guide

r12.5

Contact CA Technologies

Contact CA Support

For your convenience, CA Technologies provides one site where you can access the information that you need for your Home Office, Small Business, and Enterprise CA Technologies products. At http://ca.com/support, you can access the following resources:

- Online and telephone contact information for technical assistance and customer services
- Information about user communities and forums
- Product and documentation downloads
- CA Support policies and guidelines
- Other helpful resources appropriate for your product

Providing Feedback About Product Documentation

If you have comments or questions about CA Technologies product documentation, you can send a message to techpubs@ca.com.

To provide feedback about CA Technologies product documentation, complete our short customer survey which is available on the CA Support website at http://ca.com/docs.

Contents

Chapter 1: Overview of the Federation Manager Java SDK

This section contains the following topics:

Java SDK Functionality

The Federation Manager Java SDK is a library for interacting with an HTTP cookie that contains user identity information. The Java SDK supports the open format cookie. The open format cookie is a string of UTF-8 bytes. This format includes associated encryption algorithms designed so that information can be securely communicated between CA SiteMinder and end-user applications. Applications can be written in any common Web programming language. Using the Java SDK is not required to create an open format cookie.

Typically, CA SiteMinder sets user identity information into an HTTP cookie for consumption by an end-user application. End-user applications can use the Java SDK to extract identity information, the authentication context, name ID, and name ID format from the cookie. In addition, third-party Web access managers can create a cookie and provide user credentials to CA SiteMinder.

Java SDK Files

The Federation Manager Java SDK is implemented as several Java archive files. You specify their location during installation. The most important file, fedsdk.jar, contains the IFederationOpenIdentity.java and other supporting Java classes. Your Java application must instantiate an implementation object for one of this interface and call the methods as your requirements dictate.

Chapter 2: Installation of the Java SDK

Install the Java SDK on Windows Systems

The following procedure describes the installation on Windows platforms.

Important! You must have a Java Runtime Environment (JRE) installed on your target system. Refer to the Platform Support Matrix on the Technical Support site for the supported version.

To locate installation kits

1. Go to the Technical Support site.
2. Log on to the site.
3. Click Download Center.

Search the Download Center for the installation kit you need and download it to your local system.

To install the Federation Manager Java SDK on Windows

1. Exit all applications that are running.
2. Navigate to where the installation executable is located.
3. Double-click ca-fedmgr-java-sdk-12.51-win32.exe.

   The installation wizard starts.
4. Follow the prompts in the installation wizard.
5. After the installation is complete, reboot your system.

The installation of the Java SDK on Windows is complete.

Install the Java SDK on UNIX Systems

The Solaris and Linux operating environments support the Federation Manager Java SDK

Important! You must have a Java Runtime Environment (JRE) installed on your target system. Refer to the Platform Support Matrix on the Technical Support site for the supported version.

To locate installation kits

1. Go to the Technical Support site.
2. Log on to the site.
3. Click Download Center.

Search the Download Center for the installation kit you need and download it to your local system.

To install the Federation Manager Java SDK on UNIX

1. Exit all applications that are running.
2. Navigate to where the installation executable is located.
3. Run the binary for your platform:

   Solaris: ca-fedmgr-java-sdk-12.51-sol.bin

   Linux: ca-fedmgr-java-sdk-12.51-linux.bin

   The installation wizard starts.
4. Follow the prompts in the installation wizard and complete the installation.
5. After the installation is complete, reboot your system.

The installation of the Java SDK is complete.

Chapter 3: Federation Manager Java SDK Programming Interfaces

This section contains the following topics:

IFederationOpenIdentity Interface

The IFederationOpenIdentity interface defines methods for manipulating the federation open format cookie. The interface supports the following tasks:

- Initialize the SDK logger specific to an application.
- Extract user identity information from the cookie in an HTTP request, in a Java Cookie object, or in String format.
- Initialize values for the cookie name, domain, and security zone.
- Set a shared secret used to derive a key for cookie encryption and decryption.
- Create the open format cookie.
- Pass identity attributes to an application.
- Get and set URIs for AuthnContext and UserConsent.

To obtain an implementation of the IFederationOpenIdentity interface, call one of the implementation methods defined in the IdentityFactory. These methods require specifying a string for the cryptographic transformation of the cookie.

The following password-based encryption combinations are available for standard installations:

- PBE/SHA1/AES/CBC/PKCS12PBE-1000-128
- PBE/SHA1/AES/CBC/PKCS12PBE-1000-192
- PBE/SHA1/AES/CBC/PKCS12PBE-1000-256
- PBE/SHA256/AES/CBC/PKCS12PBE-1000-128
- PBE/SHA256/AES/CBC/PKCS12PBE-1000-192
- PBE/SHA256/AES/CBC/PKCS12PBE-1000-256
- PBE/SHA1/3DES_EDE/CBC/PKCS12PBE-1000-3
- PBE/SHA256/3DES_EDE/CBC/PKCS12PBE-1000-3

Password-based encryption (PBE) combinations are not FIPS-compatible. Any of the FIPS-mode encryption combinations listed following requires using the Java SDK to operate properly.

The following encryption combinations are FIPS-compliant and also available for standard installations:

- AES128/CBC/PKCS5Padding
- AES192/CBC/PKCS5Padding
- AES256/CBC/PKCS5Padding
- 3DESEDE/CBC/PKCS5Padding

Note: All cryptographic strings and their corresponding constant names are listed in IdentityCrypto.java.

Open Format Cookie

The federation open format cookie lets applications assert user attributes to CA SiteMinder and consume user attributes encapsulated by CA SiteMinder. The open format cookie has the following general characteristics:

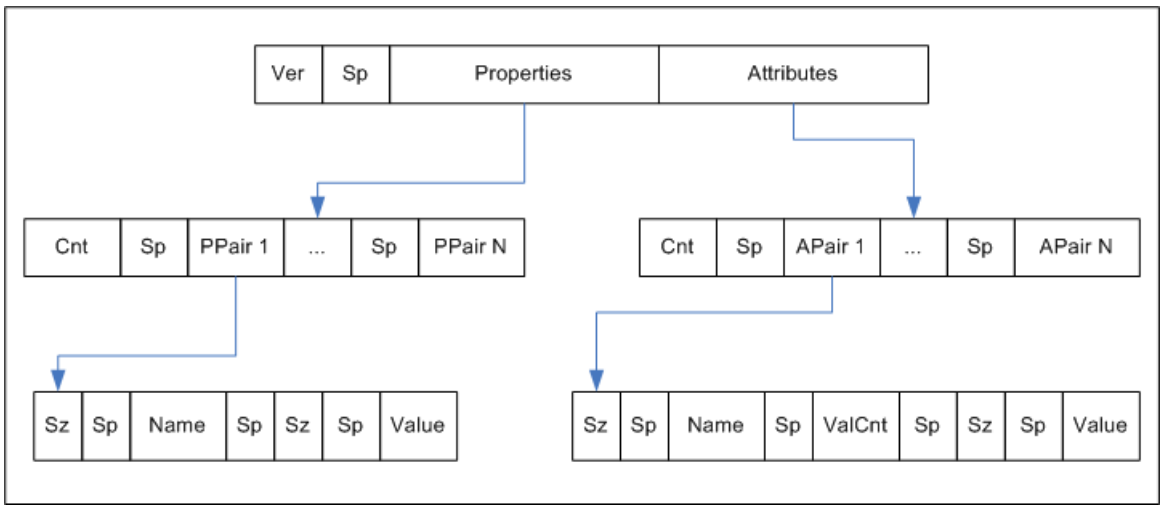
- The cookie is accessible by applications written in any programming language.
- The cookie content consists of a string of UTF-8 bytes, which supports international character sets.
- The combined size in UTF-8 bytes of each name/value pair precedes the name/value pair.
- Space characters are added for legibility.
- The cookie is simple to parse and easily extensible.

Important! If the cookie contains any unsafe characters such as '=', enclose the value in double quotes. You can specify this option through the user interface, or through the SDK.

The open format cookie contains the following property information:

- Cookie Version
- Name ID
- Name ID Format
- Session ID
- AuthnContext
- UserDN (same as User ID)

The following diagram shows the open format:



Key:

- Ver — the cookie format version; for Federation Manager r12.1, this value is 1.
- Sp — an ASCII space character, used only to improve readability
- Properties — information about the principal
- Attributes — SAML attributes from the Assertion
- Cnt — the number of name value pairs that follow, represented in ASCII
- Sz — the length of the name or value that follows
- ValCnt — the number of attribute values that follow. For Federation Manager r12.1, multiple values for an attribute are not supported. This value must be set to 1.

The Backus-Naur Form (BNF) for this format is following (0* means 0 or more; 1* means at least 1).

- DIGIT = ASCII digit (0 through 9)
- CHAR = UTF-8 character
- Sp = ASCII space (character 32)
- Token = 1*CHAR
- Cookie = Version Sp Properties Attributes
- Version = 1*DIGIT
- Cnt = 1*DIGIT
- Properties = Cnt 1*PPair
- Attributes = Cnt 0*APair
- ValCnt = 1*DIGIT
- PPair = Sz Sp Name Sp Sz Sp Value
- APair = Sz Sp Name Sp ValCnt Sp Sz Sp Value
- Sz = 1*DIGIT
- Name = Token
- Value = Token

FedSdkLogger Interface

The FedSdkLogger interface provides the following methods for specifying custom logging messages:

void logTrace (string fileName, string methodName, string msg)

Logs a trace message.

void logError (string fileName, string methodName, string msg)

Logs an error message.

Chapter 4: Using the Federation Manager Java SDK

This section contains the following topics:

Program Flow at the Relying Party Using the Open Format Cookie

A brief description of Java SDK program flow at the relying party is following.

1. The Java Application creates an implementation class of the IFederationOpenIdentity interface using the IdentityFactory interface.
2. The Java application calls the extractCookie() method to extract the cookie from the HttpServletRequest object. This method also decrypts the cookie and puts the identity attributes in the Storage Map.
3. Alternatively, the Java application can also call the processCookie() method to extract all the attributes from a cookie object and set them in the Storage Map.
4. The Java application can get values for all the attributes that are put in the Storage Map using the getAttributes(), getAttribute(), getAuthnContext(), getSessionID(), getNameID(), getNameIDFormat(), and getUserConsent() methods.
5. The Java application can set values for attributes in the cookie using the setAuthnContext() and setUserConsent() methods.
6. The Java application can determine whether the cookie is no longer valid by calling the isExpired() method, with or without specifying a skew time. The method compares the expiration time stamp on the cookie, adding in the optional skew time, with the current GMT time. If the GMT time is greater, the cookie has expired. The cookie's expiration time stamp is specified using setTimeToLive() method when the cookie is created.

See the Javadoc reference for detailed information about these methods.

Delegated Authentication Using the Open Format Cookie

Delegated authentication lets a third-party access management system authenticate a user and then share the user credentials with CA SiteMinder deployed at the asserting party. These credentials are shared either through a cookie, or in a query string.

Note: This guide discusses delegated authentication using the cookie and the Java SDK. See the *Federation Manager Guide: Partnership Federation* for information about delegated authentication using a query string.

If the third-party access manager and the asserting party intend to use a cookie to communicate the authenticated user ID, the access control application can follow these steps:

1. Implement the Federation Manager Java SDK.
2. Construct an implementation class for the IFederationOpenIdentity interface.
3. Call the createCookie method.

To construct the implementation class, the access control manager must know the cookie zone and password configured in Federation Manager. These values are communicated out-of-band. The third-party access management system must be in the same cookie domain as the asserting party.

The constructor from the IdentityFactory.java class to use when creating a cookie for delegated authentication is listed following.

```
/**
 * Gets an implementation of the IFederationOpenIdentity interface.
 *
 * @param cryptoInstance A cryptographic string; supported values are
 *  listed in IdentityCrypto.java.
 * @param bUseHmac A Boolean value that indicates whether to use HMAC.
 */
  public static IFederationOpenIdentity getInstance(cryptoInstance, bUseHmac)
```

The access control manager can encrypt the cookie itself using password-based encryption, or it can use one of the FIPS-compliant cryptographic strings. If you chose a FIPS-compliant string, use the encryption provided by the Java SDK.

Here is a code snippet example of the cookie creation:

```
IFederationOpernIdentity openID =
IdentityFactory.getInstance(IdentityCrypto.AES128, false);

String domain = ".moon.com";
String zone = "FED";
String name = "CryptoID"
String password = "";

openID.initCookieInfo(domain, zone, name, password);
```

```
openID.setLoginID = "TomJones";

openID.createCookie(HttpResponse);
```

The createCookie method uses the login ID to create a cookie value that is encrypted and added to the HttpSevletResponse object. After the request is redirected, the servlet container automatically passes the cookie.

Federation Manager Java SDK Logging

The default Java SDK logger writes messages to the standard output stream. Logging is disabled by default.

To enable Federation Manager Java SDK logging

1. Copy the sdkloggingconfig.properties file from the *sdkroo*t\sample folder and place it in any desired folder. Be sure that the folder is in the CLASSPATH.
2. Set the the value of the sdk.logging.enable parameter to Y in the sdkloggingconfig.properties file.

Logging is enabled.

Java SDK Sample Application Overview

The Java SDK sample application simulates a relying party Java application. The application consumes the cookie sent by the Federation Manager deployment running at the relying part of the federation partnership.

The sample application demonstrates how a Java application can get the cookie from the incoming request and extract user identity information and the assertion attributes that are sent to the relying party. The sample application requires that Federation Manager is installed at the relying party and is configured to redirect the user to the URL of the sample application servlet.

Java SDK Sample Application Deployment

Deployment of the Java SDK sample application requires installing Tomcat and Federation Manager at the relying party.

To deploy the Java SDK sample application

1. Install the Java SDK package at any preferred location.
2. Set the environment variable FEDSDKROOT to the installation Directory of Java SDK.

   Note: The value of FEDSDKROOT points to the location of the SDK directory. Example: C:\Program Files\CA\Federation Manager\sdk.

   This environment variable is set automatically on Windows, but must be exported manually on UNIX platforms.
3. Install Tomcat 5.0 and set the TOMCAT_HOME environment variable to point to the Tomcat root folder.

   Note: Tomcat must be installed on a different system from the one Federation Manager is installed on.
4. Deploy FEDSDKROOT\sample\javasdk/war to the Web server by copying it to the TOMCAT_HOME\webapps\ folder.
5. Start the Tomcat server.
6. Try accessing the link “http://<FQDN of Tomcat Host>:<port num>/” to determine whether Tomcat is up and running.
7. When using the open format cookie, update fedsample.properties in the TOMCAT_HOME\webapps\javasdk\WEB-INF\classes folder as follows:
   - RedirectMode is the value redirect mode. Use OPEN for the open format cookie.
   - CookieDomain is the value of cookie domain as set in the Federation Manager Create Partnership dialog.
   - CookieName is the value of the cookie name as set in the Federation Manager Create Partnership dialog.
   - CryptoInstance is the value of the encryption transformation, which is configured in the Federation Manager Create Partnership dialog.

- UseHmac specifies the value of the Enable HMAC check box as set in the Federation Manager Create Partnership dialog. Use the value no when the check box is not checked, or the value yes when the check box is checked.
- ShowAttributeMap specifies whether the data in the Assertion Mapis displayed. The value is no when data in the Assertion Map is not to be displayed/ The value is yes when all the data in the Assertion Map is to be displayed. This value is configured in the Federation Manager Create Partnership dialog. If the value is set to no, only the list of attributes mentioned in SpSideAttributeKey parameter are displayed.
- SpSideAttributeKey specifies the attribute or attributes (comma separated) from the request that are displayed.
- CharsetEncoding specifies the charset encoding of the response that is displayed on the screen.

8. The following parameters must be updated if you are testing light weight provisioning:
   - EnableProvisioningTest specifies whether light weight provisioning is enabled. Use the value no if provisioning is not enabled. Use the value yes to enable testing light-weight provisioning.
   - AssertionConsumerUrl specifies the supply Assertion Consumer URL.
   - UDType specifies odbc or ldap, depending on the User Directory type. The connection parameters associated with the type must be specified.
9. Update the sdkloggingconfig.properties file to enable logging. Logging is disabled by default.
10. Install Federation Manager at the relying party of a Federation partnership and define an asserting party-relying party partnership.
    a. Select appropriate Redirect mode.
    b. Specify the Target URL of the partnership. Enter one of the following:
       - The URL of the SDK Sample App, such as http://<FQDN of target machine>:<Tomcat port>/javasdk/SpSideAttributeServlet.
       - The url of the relying party, http://<FQDN of SP>:CA Portal and in proxyrules.xml(location: %FEDROOT%\ proxy-engine\conf\proxyrules.xml make the entry http://<FQDN of target machine>:<Tomcat port>/javasdk/SpSideAttributeServlet.

The sample application is now deployed and ready to run.

Java SDK Sample Application Execution

After you have installed Tomcat and Federation Manager, you can run the Java SDK sample application.

To run the Java SDK sample application

1. Start the Tomcat Server where the sample application is deployed:
   - On Windows, use the services control panel.
   - On UNIX, use the startup script of Tomcat.
2. Run the configured federation transaction to redirect to the SDK Sample Application.

   The sample application decodes the legacy cookie and displays the user identity information contained in the cookie.

Java SDK Sample Application Customization

The sample application can be modified using build.bat or build.sh scripts to regenerate the fedsdksample.jar.

To customize the sample Java application

1. Modify SpSideServlet or SpSideAttributeServlet.java as desired.
2. Verify that the JDK is installed and JAVA_HOME is set appropriately for your JDK installation.
3. Run build.bat (Windows) or build.sh (UNIX) to build the fedsdksample.jar file.

The customized version of the sample application is ready to run.

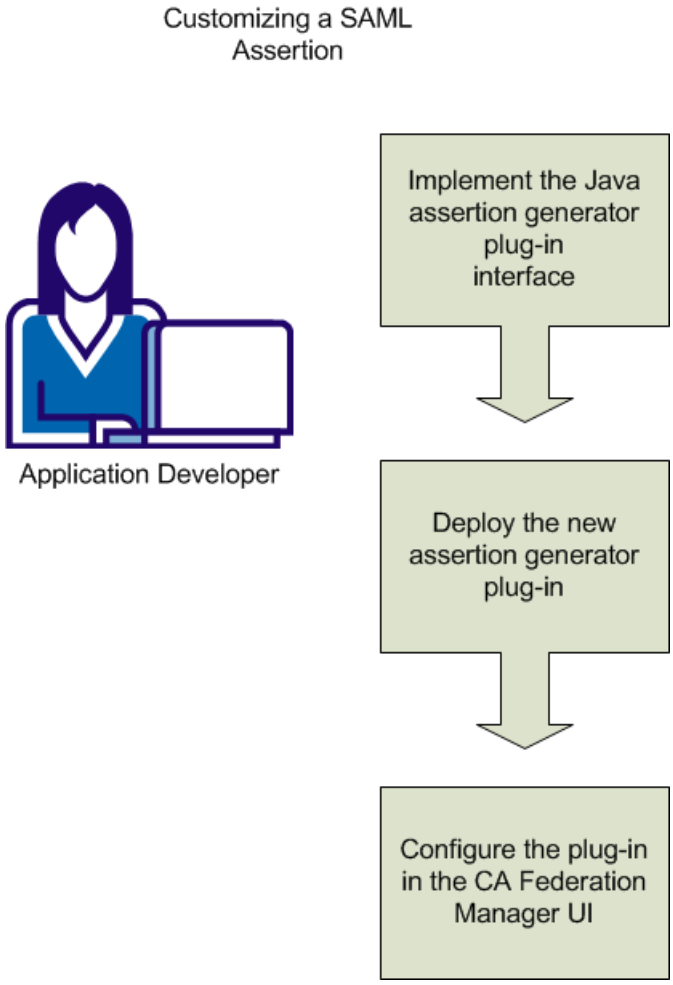
Customizing a SAML Assertion

Security domains exchange authentication and authorization using data packages named assertions. The Security Assertion Markup Language (SAML) is an open standard that specifies the format of an assertion. A federated partnership consists of an identity provider (producer of an assertion) and a service provider (consumer of an assertion).

An enterprise can modify the content of an assertion based on the business agreements between the federated partners. For example, one partner can require user-friendly name equivalents for the attributes in the assertion. Or, a partner can opt to include the XML-type designation for each attribute in the assertion.

Federation Manager creates SAML assertions with its implementation of the AssertionGeneratorPlugin.java interface. An Application Developer can enhance the contents of the SAML assertion by overwriting the existing implementation class.

The diagram shown following illustrates the process of creating a custom assertion generator plug-in.



The process of customizing a SAML assertion includes these steps:

1. Implement the Java assertion generator plug-in interface (see page 24).
2. Deploy the new assertion generator plug-in .
3. Configure the assertion generator plug-in in the Federation Manager UI.

Implement the Java Assertion Generator Plug-in Interface

You create a custom assertion generator plug-in by implementing the AssertionGeneratorPlugin.java interface. The minimum requirements for the implementation class are listed following.

Follow these steps:

1. Provide a public default constructor method that contains no parameters.
2. Provide code that helps ensure that the implementation is stateless, so that many threads can use a single plug-in class.
3. Include a call to the customizeAssertion method.

   Example

   In this example, imagine that the application developer defines handler.updateNameID to create user-friendly name attributes.

```
/**
     * <p>Performs Assertion Generator callout functionality to customize the
     * SAML assertion in the <code>AssertionGeneratorPlugin</code> object and
     * returns a result.</p>
     * @param apiContext    A context object that provides methods for sending
     *                     log, trace, and error messages to the Policy Server.
Use the APIContext.getAttrMap() method to retrieve attributes posted by the
application specified in the Application URL.
     * @param userContext   A context object that allows a custom object to set
     *                       and retrieve information about a user in a user
     *                       directory. The information includes user
     *                       attributes and directory attributes associated
     *                       with the user.
     * @param pluginParam    The string for Assertion plug-in parameters.
     * @param inputAssertion    The current XML token representing the SAML
Assertion.
     * @param outputAssertion   The final XML token representing the SAML
Assertion.
     * @return  0 if assertion is customized successfully, or -1 if no
customization or an error occurred.
     * If the method fails, the outputAssertion is ignored.
     * @throws java.lang.Exception For cases when the customization terminates
unexpectedly.
     *
     **/
```

```
    public int customizeAssertion(APIContext apiContext, UserContext
userContext,String pluginParam,
     String inputAssertion, final StringBuffer outputAssertion) throws Exception
{
        if (inputAssertion == null || inputAssertion.equals("")) {
            // Indicates non-zero for an error.
            apiContext.trace(PLUGIN_TAG, "Received null or empty response for
customization");
            return -1;
        }
        apiContext.trace(PLUGIN_TAG, "Entering customizeAssertion");
        StringBuffer newAssertion = new StringBuffer(inputAssertion);

        try
        {
            Saml1AssertionHandler handler =
                initHandler(apiContext, userContext);
            handler.updateNameID(newAssertion);
            handler.addAttributes(pluginParam, newAssertion);
        }
        catch(Throwable th)
        {
            apiContext.error("SAML1AssertionSample: " + th.getMessage());
            StringWriter writer = new StringWriter();
            th.printStackTrace(new PrintWriter(writer));
            writer.flush();
            apiContext.trace(PLUGIN_TAG,
                             "Error customizing Assertion:\n" +
writer.toString());

            apiContext.trace(PLUGIN_TAG, "Done customizeAssertion");
            return -1;
        }

        outputAssertion.append(newAssertion);

        apiContext.trace(PLUGIN_TAG, "Done customizeAssertion");

        // return "success"
        return 0;
    }
```

Note: The syntax requirements and use of the parameter string that is passed into the customizeAssertion method is the responsibility of the custom object.

Deploy the Assertion Generator Plug-in

After you have coded your implementation class for the AssertionGeneratoPlugin.java interface, compile it and verify that CA SiteMinder can find your executable file.

Follow these steps:

1. Compile the assertion generator plug-in code in one of the following ways:
   - If you are using a sample plug-in, use the build script to compile the plug-in. The build scripts are installed in the directory *federation_mgr_sdk_home*\sample. The build scripts are:

     Windows: build_plugin.bat

     UNIX: build_plugin.sh

     A compiled sample plug-in, fedpluginsample.jar, is in the directory *federation_mgr_sdk_home*\jar.
   - If you write your own plug-in, include smapi.jar when you compile your plug-in.
2. In the JVMOptions.txt file, modify the -Djava.class.path value so it includes the classpath for the plug-in. Locate the JVMOptions.txt file in the directory *federation_mgr_home*\siteminder\config.

   You can place the plug-in jar in any directory and have the JVMOptions.txt file point to it. To use the sample plug-in, modify the classpath to point to fedpluginsample.jar; however, do not modify the classpath for smapi.jar.

   Note: To use Apache Xerces or Xalan in your plug-in, use the Xerces or Xalan binary files installed with CA SiteMinder. The binaries are not installed with the Federation Manager SDK. Using these files is necessary for compatibility reasons.

Configure the Assertion Generator Plug-in in the UI

To configure the assertion generator plug-in, you provide values for settings in the Administrative UI.

Note: Do not configure the plug-in settings until you deploy the plug-in.

Follow these steps:

1. Log on to the Administrative UI.
2. Navigate to the Assertion Configuration step of the Partnership wizard for the partnership you want to modify.

3. Enter values for the following fields:

   Plug-in Class

   Specifies the Java class name of the plug-in. Enter a name. This plug-in is invoked at run time.

   Example: com.mycompany.assertiongenerator.AssertionSample

   The plug-in class can parse and modify the assertion, and then return the result to CA SiteMinder for final processing. Specify an assertion generator plug-in for each relying party. A compiled sample plug-in is included in the directory *federation_mgr_sdk_home*/jar.

   Plug-in Parameter

   (Optional). Specifies the string that CA SiteMinder passes to the plug-in as a parameter at run time. The string can contain any value; there is no specific syntax to follow.

   The plug-in interprets the parameters that it receives. For example, the parameter is the name of an attribute, or the string can contain an integer that instructs the plug-in to do something.

The assertion generator plug-in is coded, compiled, and in place. The CA SiteMinder assertion generator creates enhanced assertions as defined by the federation partners.

Index