CA Federation Manager

.NET 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 .NET SDK

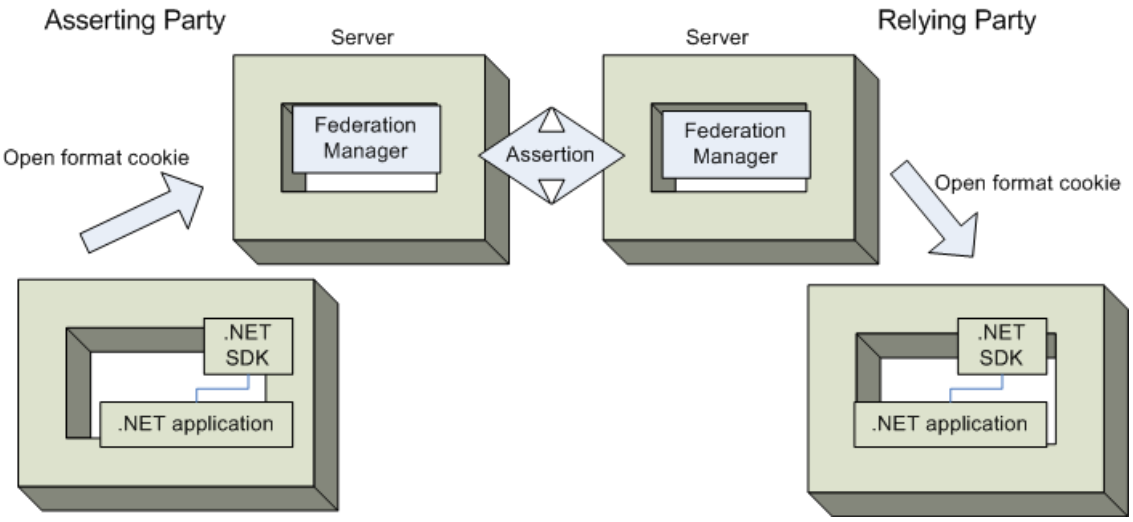
This section contains the following topics:

Architecture of the .NET SDK

The Federation Manager .NET SDK helps a .NET application to federate. Using the .NET SDK, .NET applications can provide user information to Federation Manager, and can consume user information provided by Federation Manager. The .NET SDK uses a global open format cookie to represent user identity and encapsulate the user principal and attributes. The .NET SDK uses a key derived from a shared secret to encrypt the cookie. Any application that knows the shared secret and the cryptographic transform can consume the cookie and retrieve user information. The .NET SDK uses the AES algorithm for encrypting and decrypting the open format cookie.

A .NET application on the asserting party side uses the .NET SDK to pass the login ID for authenticated users to Federation Manager. Federation Manager extracts the login ID from the cookie and adds it to a Federation Assertion, which is sent to relying party. Federation Manager can add additional attributes to the cookie and change some of the cookie settings, for example, the maximum age for a cookie. A .NET application on the relying party side uses the .NET SDK to retrieve user and session-related information sent by Federation Manager.

The following diagram shows the role of the .NET SDK at the asserting party and the relying party:



Programming Prerequisites

The .NET SDK is implemented in C#, only using features that are part of the Microsoft Common Language Specification (CLS). The .NET SDK is therefore accessible from applications written in any language that supports the CLS, for example, Visual Basic .NET, Visual C# .NET, and Visual C++ .NET.

The .NET SDK interfaces are available through the CA.Federation.FedIdentitySdk.dll. .NET applications can reference this DLL using the namespace CA.Federation.FedIdentitySdk.

The .NET application has to pass cookie zone, cookie name, and the shared secret to the .NET SDK. The .NET application can store this data in any way convenient, for example, in a configuration file. The application can encrypt the password, but must decrypt it before passing it to the .NET SDK. The password must be passed as a plain text character array. The configuration values of cookie zone, cookie name, and encryption password must be the same at both the sides (the .NET Application and Federation Manager). These values are communicated out-of-band.

Chapter 2: Installation of the .NET SDK

This section contains the following topics:

Install the .NET SDK on Windows

The installation of the CA Federation Manager .NET SDK is fully automated. The installation program guides you through the process.

Important! You must have version 3.5 of the .NET Framework installed on the system where you are installing the .NET SDK; otherwise, the installation fails. Supported operating systems are listed in the Compatibility Matrix on the Technical Support site.

You can specify where the .NET is installed. The link library, CA.Federation.FedIdentitySdk.dll, by default is installed in C:\Program Files\CA\Federation Manager\sdk\dotnet\bin.

You run the .NET SDK installer with ca-fedmgr-dotnet-sdk-12.51-win32.exe. The executable is located on the Technical Support site.

To locate installation kits on the Support site

1. Click Technical Support.
2. Log into CA Support Online.
3. Click Download Center.

Search the Download Center for the appropriate installation kit.

Follow these steps:

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

   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 Federation Manager Windows Agent is complete.

Chapter 3: .NET SDK Components

This section contains the following topics:

Open Format Cookie

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

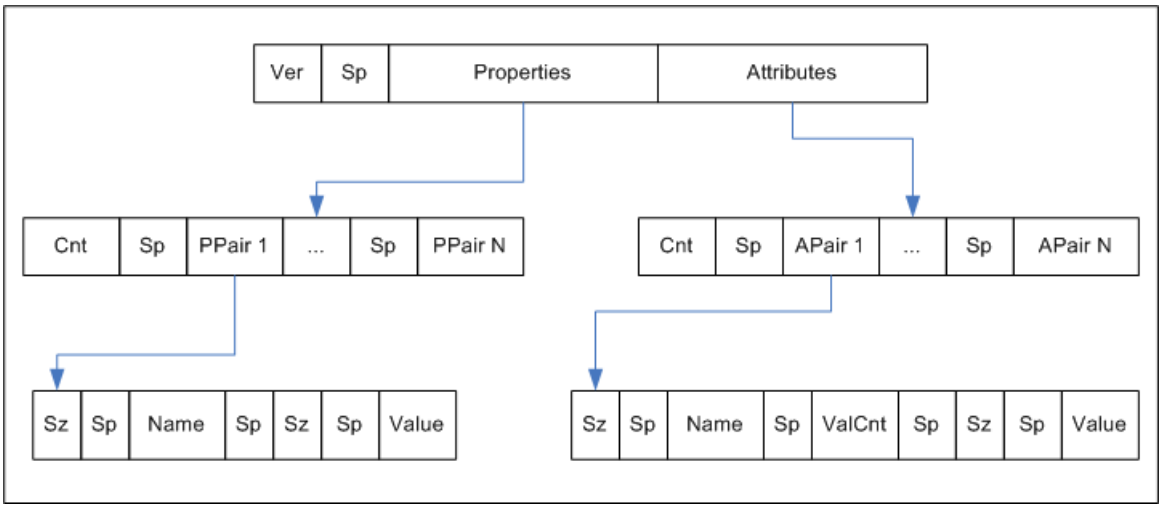
- The cookie is accessible by applications written in any programming language.
- The cookie content consists of a string of UTF-8 bytes, which supports international character sets.
- The combined size in UTF-8 bytes of each name/value pair precedes the name/value pair.
- Space characters are added 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)
- UserConsent
- Login ID
- ExpiresON (expiration time)

The following diagram shows the open format:



Key:

- Ver — the cookie format version. 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

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

IFederationOpenIdentity Interface

The IFederationOpenIdentity interface defines methods for manipulating the open format cookie. The classes exposed by .NET SDK are available under the namespace CA.Federation.FedIdentitySdk. You implement the IFederationOpenIdentity interface by calling one of the methods from the IdentityFactory class.

See the Doxygen-generated reference for detailed information about this interface.

Identity Factory

The IdentityFactory class provides methods for obtaining an implementation of the IFederationOpentIdentity interface.

Note: The only supported cryptographic transformation is "AES128/CBC/PKCS5Padding". You can also use NULL to get the default.

The IdentityFactory class includes the following methods:

static IFederationOpenIdentity GetInstance (string cryptoInstance)

Generates an implementation object of the IFederationOpenIdentity interface.

static IFederationOpenIdentity GetInstance (string cryptoInstance, bool bUseHmac)

Generates an implementation object of the IFederationOpenIdentity interface.

static IFederationOpenIdentity GetInstance (string zoneName, char[] password, string domain, string cryptoInstance)

Generates an implementation object of the IFederationOpenIdentity interface.

static IFederationOpenIdentity GetInstance (string zoneName, char[] password, string domain, string cryptoInstance, bool bUseHmac)

Generates an implementation object of the IFederationOpenIdentity interface.

IFedIdentitySDKLogger Interface

The IFedIdentitySDKLogger interface provides the following methods for specifying custom logging messages

void LogTrace (string fileName, string methodName, string message)

Logs a trace message.

void LogError (string fileName, string methodName, string message)

Logs an error message.

Chapter 4: Using the .NET SDK

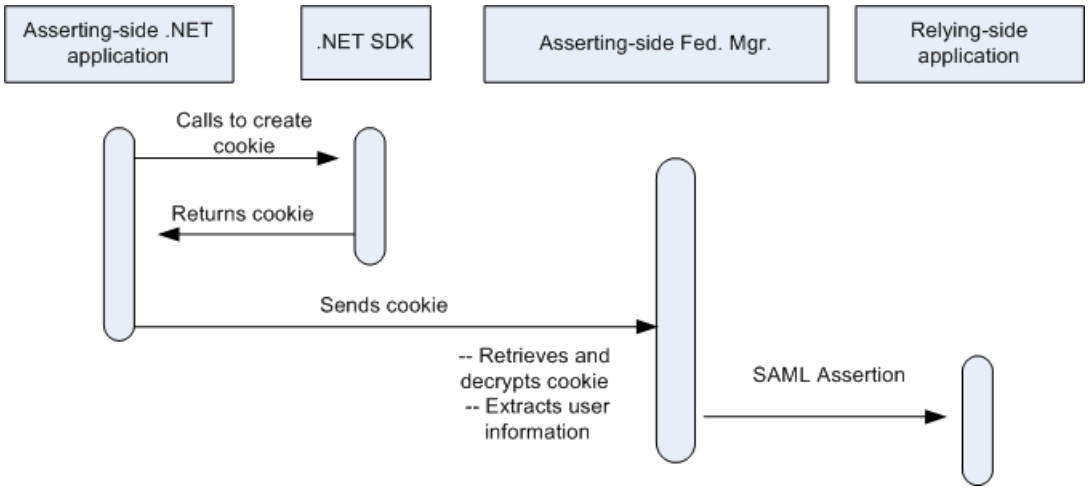
This section contains the following topics:

Program Flow at the Asserting Party

With Federation Manager at the asserting party, a .NET application can provide Federation Manager with user identity information. Program flow with Federation Manager at the asserting party proceeds as follows:

1. The .NET application calls the .NET SDK to generate an open format cookie with identity information.
2. The .NET SDK returns an encrypted cookie. The key used to encrypt the cookie is derived from a shared secret, communicated between Federation Manager and the application out-of band.
3. The .NET application sends the cookie to Federation Manager at the asserting party.
4. Federation Manager receives and decrypts the cookie.
5. Federation Manager extracts user identity information from the cookie.
6. Optionally, Federation Manager can modify the cookie by updating or adding attributes.
7. Federation Manager inserts the user identity information into a SAML Assertion.

The following diagram shows program flow at the asserting party:

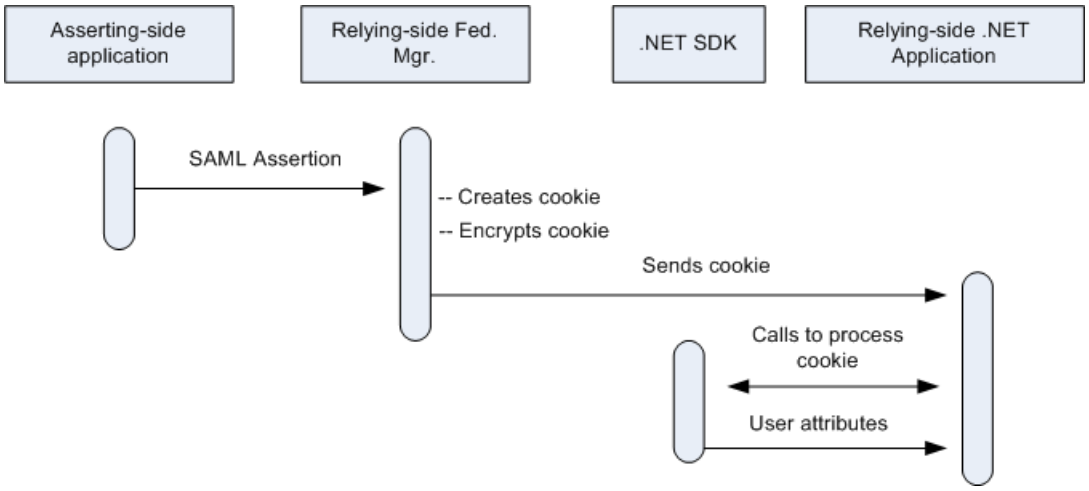


Program Flow at the Relying Party

With Federation Manager at the relying party, the .NET application can receive user information from Federation Manager. Program flow with Federation Manager at the relying party proceeds as follows:

1. Federation Manager receives a SAML Assertion during request processing.
2. Federation Manager creates the cookie with the latest user information.
3. Federation Manager encrypts the cookie using a FIPS-compliant algorithm. The key used to encrypt the cookie is derived from a shared secret, communicated between Federation Manager and the application out-of band.
4. Federation Manager sends the encrypted open format cookie to the .NET application.
5. The .NET application calls the .NET SDK to decrypt and process the cookie.
6. The .NET application retrieves values for assertion attributes and principal attributes.
7. The .NET 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.
8. The .NET application can also set URIs for AuthnContext and UserConsent.

The following diagram shows program flow at the relying party:



Federation Manager .NET SDK Logging

When enabled, .NET SDK logger writes messages to the standard output stream. Logging is disabled by default.

To enable Federation Manager .NET SDK logging

1. Copy the Logger.xml file from the *.NET SDK Installation directory*\config and place it with the .NET SDK DLL the \bin folder.
2. Set the EnableLogging parameter to yes in Logger.xml.

Logging is enabled.

Programming Examples

The following code fragments illustrate creating an open format cookie:

// Gets an object reference of the interface type IFederationOpenIdentity, bound to a custom
// implementation of the IFederationOpenIdentity interface.
// AES128/CBC/PKCS5Padding is the only supported cryptographic transformation string.

```
IFederationOpenIdentity openID =
IdentityFactory.GetInstance("AES128/CBC/PKCS5Padding", UseHMACFlag);
```

// Initializes the parameters required to create cookie.

```
openID.InitCookieInfo(Domain, CookieZone, CookieName, Password);
```

// Sets a user attribute.

```
openID.LoginID = txtLoginID.Text;
```

// Creates an open format cookie and sets it into the response object.

```
openID.CreateCookie(HttpResponse);
```

The following code fragments illustrate consuming an open format cookie:

// Gets an object reference of the interface type IFederationOpenIdentity, bound to a custom
// implementation of the IFederationOpenIdentity interface.
// AES128/CBC/PKCS5Padding is the only supported cryptographic transformation string.

```
IFederationOpenIdentity openID =
IdentityFactory.GetInstance("AES128/CBC/PKCS5Padding", UseHMACFlag);
```

// Initializes parameters needed to extract cookie.

```
openID.InitCookieInfo(Domain, CookieZone, CookieName, Password);
```

// Extracts the cookie from the HttpRequest, decrypts it, and saves the attributes in a Hashtable.

```
openID.ExtractCookie(HttpRequest);
```

// Retrieves some attributes.

```
String id = openID.LoginID;
```

```
String nid = openID.NameID;
```

.NET SDK Sample Application

The .NET test application generates an open format cookie and consumes it using the.NET SDK. The test application can be deployed in a number of ways. One suggested approach is listed following.

Note: Make sure the IIS Web Server is set to allow ASP .NET content.

To deploy the .NET SDK test application

1. Create a folder (in this example, TestApplication).
2. Copy the following files from the *dotNet_SDK_home*\testapp to your TestApplication folder:
   - OpenCookieConsumer.aspx.cs
   - OpenCookieConsumer.aspx
   - OpenCookieConsumetUseHMAC.aspx.cs
   - OpenCookieConsumetUseHMAC.aspx
   - OpenCookieGenerator.aspx.cs
   - OpenCookieGenerator.aspx
   - web.config
3. Create a bin folder in the TestApplication directory.
4. Copy CA.Federation.FedIdentitySdk.dll from *dotNet_SDK_home*\bin to your TestApplication\bin.
5. Open the web.config file to edit. In the <appSettings> section, change the Password, Zone, and Name keys.
   - Password is the shared secret used to derive a cryptographic key
   - Zone is the cookie zone.
   - Name is the cookie name. The final name of the cookie generated includes the Zone and the Name.
6. Go to the Internet Information Services Manager.
7. Right click websites.
8. Enter a description for the Web site.
9. Assign a TCP port to the Web site (for example, 100).
10. Enter or browse the path to the Web site home directory, that is, the location of the Test Application directory.
11. On the Website Access Permissions dialog, select the Read and Run scripts (such as ASP) options.
12. Select Finish.

13. Restart IIS.
14. Access the .NET SDK Test Application open format cookie creation page.
15. Enter the login ID.
16. Click Go.

    The system displays the .NET SDK Test Application Open Format Cookie consumption page. The OpenCookieConsumer.aspx page displays the contents of the cookie. In this case, the only attribute in the cookie is Login ID.
17. Access the .NET SDK Test Application Open Format Cookie consumption page, which decrypts the open format cookie and display the principal and assertion attributes contained in the cookie.

Index