PingFederate®

.NET Integration Kit

Version 2.4

User Guide

PingIdentity®

PingFederate .NET Integration Kit *User Guide*
Version 2.4
March, 2010

Ping Identity Corporation
1099 18th Street, Suite 2950
Denver, CO 80202
U.S.A.

Phone: 877.898.2905 (+1 303.468.2882 outside North America)
Fax: 303.468.2909
Web Site: www.pingidentity.com

Trademarks

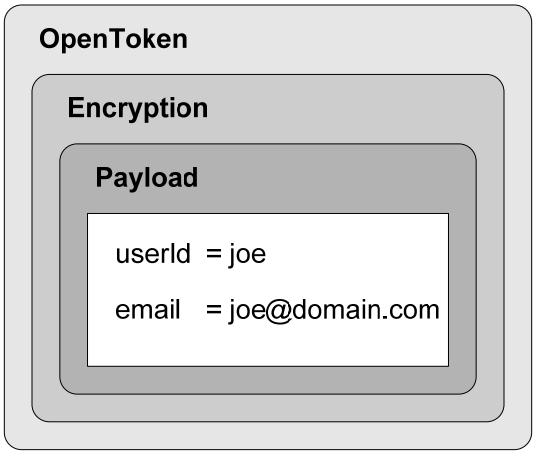
Disclaimer

Contents

Introduction

The .NET Integration Kit includes the OpenToken Adapter and a .NET agent, which allows developers to integrate their .NET applications with a PingFederate server acting as either an Identity Provider (IdP) or a Service Provider (SP). The kit allows an IdP server to receive user attributes from a .NET IdP application. On the SP side, the kit allows a .NET SP application to receive user attributes from the SP server.

The .NET Integration Kit uses an open-standard, secure token called `OpenToken` to pass user information between an application and PingFederate. The `OpenToken` is passed through the user's browser as a URL query parameter or an HTTP cookie. The data within the `OpenToken` is a set of key/value pairs, and the data is encrypted using common encryption algorithms, as illustrated below:



A specification for the OpenToken standard has been submitted to the Internet Engineering Task Force (IETF): http://tools.ietf.org/html/draft-smith-opentoken-02.

Intended Audience

This document is intended for PingFederate administrators and Web application developers who will customize one or more .NET applications to communicate with PingFederate.

We recommend that you review the PingFederate *Administrator's Manual*—specifically the information on adapters and integration kits. You should have an understanding of how PingFederate uses adapters and how they are configured. It would also be helpful for you to complete the tasks in the *.NET Sample Application Startup Guide* to have a working example (see the "Testing" section on page 14 of this document).

ZIP Manifest

The distribution ZIP file for the .NET Integration Kit contains the following:

- `/dist`
  - `opentoken-adapter-2.4.jar` – OpenToken Adapter JAR file for PingFederate server
  - `opentoken-agent.dll` – the Agent Toolkit for .NET 2.0

- `/docs`
  - `DotNet_Integration_Kit_User_Guide.pdf`
  - `DotNet_Sample_Application_Startup_Guide.pdf`
  - `DotNet_Integration_Kit_Qualification_Statement.pdf`
  - `Legal.pdf`
  - `opentoken-dotnet.chm – Agent Toolkit API Documentation`
- `/sample` – contains the .NET sample application:
  - `/IdpSample –` virtual directory for the IdP Sample Application
  - `/SpSample –` virtual directory for the SP Sample Application
  - `data.zip –` PingFederate configuration archive for the Sample Applications

System Requirements

The following software must be installed in order to implement the .NET Integration Kit:

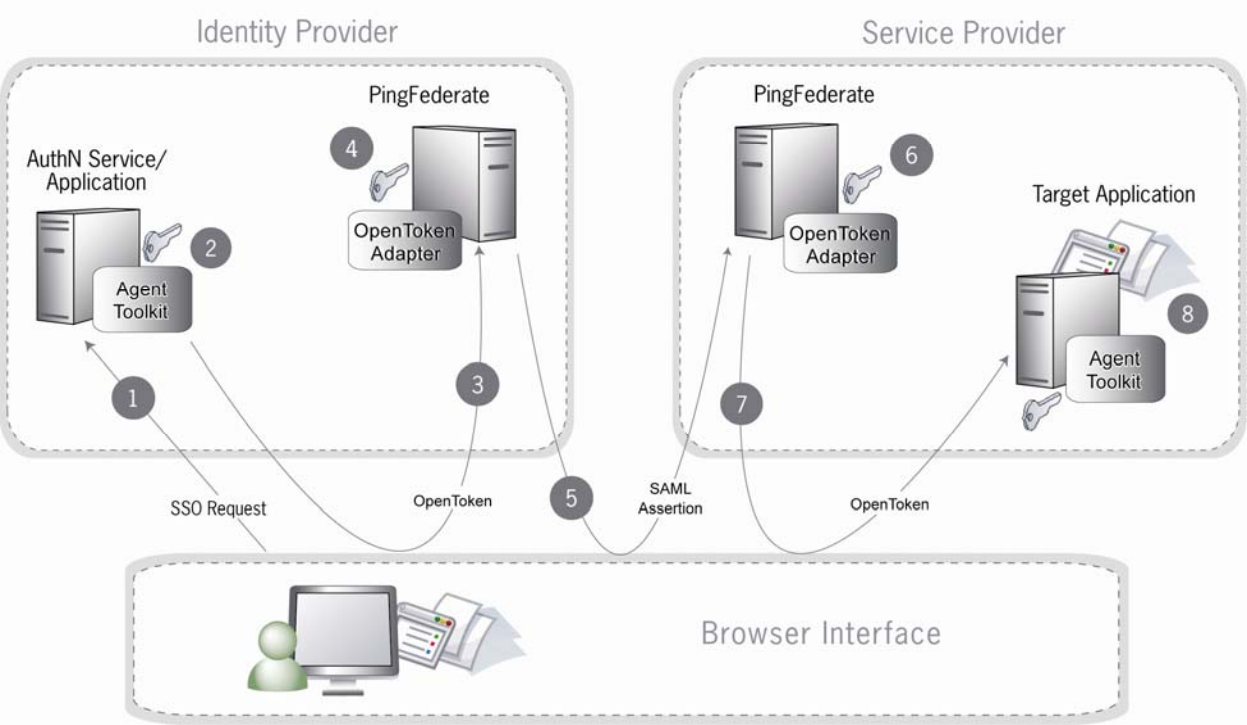
- PingFederate 5.x (or higher)
- Microsoft .NET Framework 2.0 for the agent application

Processing Overview

The .NET Integration Kit consists of two parts:

- The OpenToken Adapter, which runs within the PingFederate server
- The Agent Toolkit for .NET, which resides within the .NET application

The following figure shows a basic IdP-initiated single sign-on (SSO) scenario in which PingFederate federation servers using the .NET Integration Kit exist on both sides of the identity federation:



Sequence

1. A user initiates an SSO transaction.
2. The IdP application inserts user attributes into the Agent Toolkit for .NET, which encrypts the data internally and generates an `OpenToken`.
3. A request containing the `OpenToken` is redirected to the PingFederate IdP server.
4. The server invokes the OpenToken IdP Adapter, which retrieves the `OpenToken`, decrypts, parses, and passes the user attributes to the PingFederate IdP server. The PingFederate IdP server then generates a Security Assertion Markup Language (SAML) assertion.
5. The SAML assertion is sent to the SP site.
6. The PingFederate SP server parses the SAML assertion and passes the user attributes to the OpenToken SP Adapter. The Adapter encrypts the data internally and generates an `OpenToken`.
7. A request containing the `OpenToken` is redirected to the SP application.
8. The Agent Toolkit for .NET decrypts and parses the `OpenToken` and makes the user attributes available to the SP Application.

Note: PingFederate can be configured to look up additional attributes from either an IdP or SP data store. See the PingFederate *Administrator's Manual* for more information.

Installation and Setup

To install the .NET Integration Kit:

1. For PingFederate 6.2 and previous versions, delete the existing OpenToken Adapter from the directory:

   `<PF_install>\pingfederate\server\default\deploy`

   The adapter JAR file is `opentoken-adapter-x.x.jar`.

   Note: If the adapter JAR filename indicates version 2.1 or less, also delete the supporting library `opentoken-java-1.x.jar` from same directory.

2. If you are using PingFederate 5.1.0, delete the file opentoken-adapter.jar from the directory:

   `<PF_install>\pingfederate\server\default\lib`

3. Copy `opentoken-adapter-2.4.jar` from this distribution to the directory:

   `<PF_install>\pingfederate\server\default\deploy`

4. Start or restart the PingFederate server.

5. Configure an instance of the OpenToken Adapter.

   For detailed instructions, see the PingFederate *Administrator's Manual*.

6. On the Actions screen in the adapter setup steps, click the **Invoke Download** link and then click **Export** to download the `agent-config.txt` properties to a directory that is readable by the Agent Toolkit for .NET.

7. Once the adapter is configured, create a connection to your partner using that adapter instance. (For more information, see the "Identity Provider Configuration" or "Service Provider Configuration" chapters in the PingFederate *Administrator's Manual*.)

8. Copy the Agent Toolkit for .NET (`opentoken-agent.dll`) to the .NET application path.

Using the Agent API

Use the Agent API to read or write an `OpenToken` directly. The API allows applications to write an `OpenToken` to a given HTTP response.

Sample Code

Instantiating the agent object is done simply by invoking a constructor and loading the configuration, as in the example below:

```
using System;
using System.Collections.Generic;
using System.Text;
using opentoken;
using System.IO;
```

```
using opentoken.util;
using System.Collections;
using System.Collections.Generic;
. . . .
Agent agent = new Agent( "<PATH_TO_FILE>/agent-config.txt");
```

When the agent object is instantiated, it uses the `agent-config.txt` file to find the configuration data exported from the PingFederate OpenToken adapter instance. This configuration data includes the name of the cookie that the agent object will write, as well as the key to use when encrypting a new `OpenToken`. If the `agent-config.txt` file is not found, the agent constructor will throw an exception.

Integrating with an IdP PingFederate Server

This section provides implementation guidelines and code examples for .NET developers, covering the following types of IdP SAML 2.0 implementation profiles:

- IdP Single Sign-On (SSO)
- IdP Single Logout (SLO)

IdP Single Sign-On (SSO)

When PingFederate is configured as an IdP, it needs to be able to identify a user prior to issuing a SAML assertion for that user. When using the OpenToken Adapter with PingFederate, this means that the PingFederate server attempts to read a cookie or query parameter containing an `OpenToken` and then use the values within to identify the user. The application that starts the SSO must include an `OpenToken` so that PingFederate can identify the user. Use the Agent API to write an `OpenToken`. The API is a .NET object that provides access to functionality for writing an `OpenToken` to a given HTTP response.

The `writeToken` method takes a `System.Collections.IDictionary` collection of attributes and encodes them into an `OpenToken`, which is then written to the HTTP response.

Note: The collection of attributes *must* contain a key named "subject" in order for a valid token to be generated.

If any errors are encountered while creating the token or writing the token out to the response, a `TokenException` is thrown. The code snippet below demonstrates the use of the `writeToken` method:

```
IDictionary userInfo = new Dictionary<String, String>();
// Add userId for the logged on user as the token subject
userInfo.Add(Agent.TOKEN_SUBJECT, <userId>);
String returnUrl = "https://<PingFederate DNS>:9031" + Request["resume"];
. . . .
try {
   UrlHelper urlHelper = new UrlHelper(returnUrl);
   //see "Using the Agent API" section for sample code
   //that instantiates and configures an Agent instance
   agent.WriteToken(userInfo,Response,urlHelper,false);
```

```
   returnUrl = urlHelper.ToString();
}
catch(TokenException e) {
   // Handle exception
}
```

Passing Multi-Value Attributes

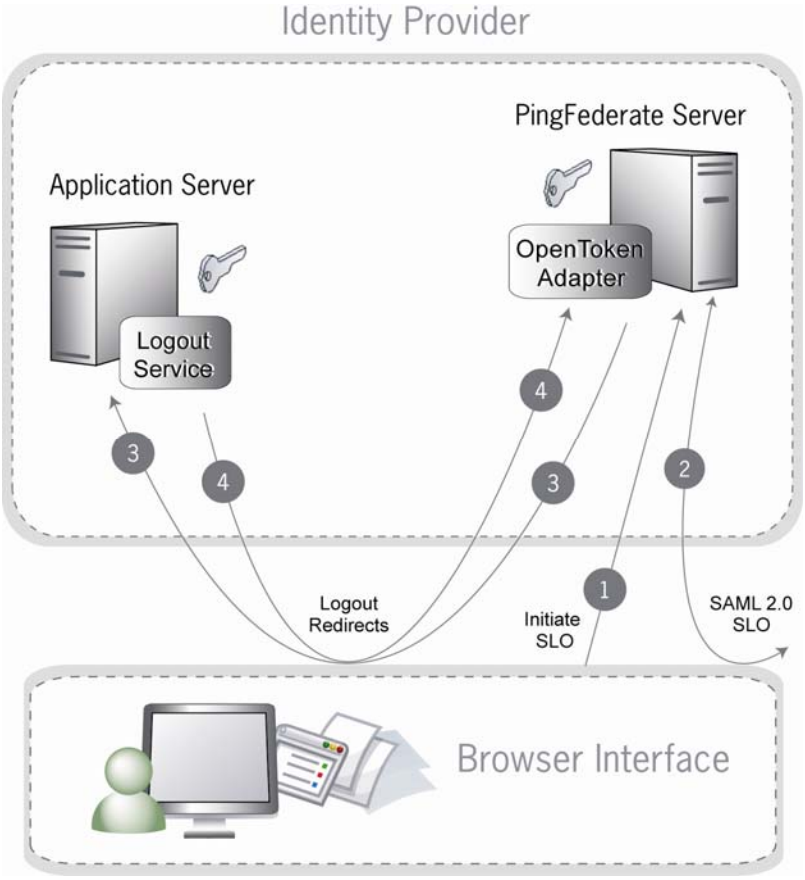
The Agent Toolkit for .NET supports passing multi-value attributes to PingFederate that will each appear in its own discrete `<AttributeValue>` element in the SAML 2.0 assertion. Multi-value attributes are passed using the `opentoken.MultiStringDictionary` collection. The following code snippet demonstrates how to pass multi-value attributes using the Agent Toolkit:

```
MultiStringDictionary userInfo = new MultiStringDictionary();
// Add userId for the logged on user as the token subject
userInfo.Add(Agent.TOKEN_SUBJECT, <userId>);
// Add an attribute GROUP with multiple values
userInfo.Add("GROUP", "Administrators");
userInfo.Add("GROUP", "Users");
String returnUrl = "https://<PingFederate DNS>:9031" + Request["resume"];
. . . .
try {
   UrlHelper urlHelper = new UrlHelper(returnUrl);
   //see "Using the Agent API" section for sample code
   //that instantiates and configures an Agent instance
   agent.WriteToken(userInfo,Response,urlHelper,false);
   returnUrl = urlHelper.ToString();
}
catch(TokenException e) {
   // Handle exception
}
```

IdP Single Logout (SLO)

When an IdP PingFederate server receives a request for SLO, it redirects the user's browser to the Logout Service defined in the IdP OpenToken Adapter configuration. The redirect URL includes an `OpenToken` containing the user attributes defined in the IdP OpenToken Adapter instance for the partner connection. The Logout Service should remove the user's session on the application server and redirect the user's browser back to the IdP PingFederate server.

The following diagram shows the flow of IdP-initiated SLO, but the architecture would also support SP-initiated SLO.



Sequence

1. User initiates a single logout request. The request targets the PingFederate server's `/idp/startSLO.ping` endpoint.
2. PingFederate sends a logout requests and receives responses for all SPs registered for the current SSO session.
3. PingFederate redirects the request to the IdP Web application's Logout Service, which identifies and removes the user's session locally.
4. The application Logout Service redirects back to PingFederate to display a logout-success page.

Below is an example code snippet for processing a logout request and sending it back to PingFederate through the user's browser:

```
// Remove local session
. . . .
IDictionary userInfo = new Dictionary<String, String>();
// Add userId for the logged on user as the token subject
userInfo.Add(Agent.TOKEN_SUBJECT, <userId>);
String returnUrl = "https://<PingFederate DNS>:9031" + Request["resume"];
Response.Redirect(returnUrl);
```

Integrating with an SP PingFederate Server

This section provides implementation guidelines and code examples for .NET developers, covering the following types of SP SAML 2.0 implementation profiles:

- SP Single Sign On (SSO)
- SP Single Sign-On (Using Account Linking)
- SP Single Logout (SLO)

SP Single Sign-On (SSO)

When PingFederate is configured as an SP, it takes inbound SAML assertions and converts them to some local format (cookie or otherwise) that can be used by an application to create a user's session. For an `OpenToken`, the PingFederate adapter takes the attributes and values from the SAML assertion and stores them in an `OpenToken` cookie or query parameter in the user's browser. The user is then redirected to the target application, which can then identify the user from the `OpenToken`, using the `Agent` API.

As with the IdP, you can use the `Agent` API to read tokens directly. The `Agent` API is a .NET class that provides access to functionality for reading an `OpenToken` from a given HTTP request.

The `readToken` method inspects the cookie (or query parameters, depending on the configuration of the agent instance) and decodes the `OpenToken`, returning a collection of attributes or `null` if no token is found or an error is encountered. In the case of an error, a `TokenException` is thrown. The following code demonstrates the use of this method:

```
try {
   //see "Using the Agent API" section for sample code
   //that instantiates and configures an Agent instance
   IDictionary userInfo = agent.ReadToken(Request);
   if(userInfo != null) {
      String username = (String)userInfo[Agent.TOKEN_SUBJECT];
   }
}
catch(TokenException e) {
   // Handle exception
}
```

Receiving Multi-valued Attributes

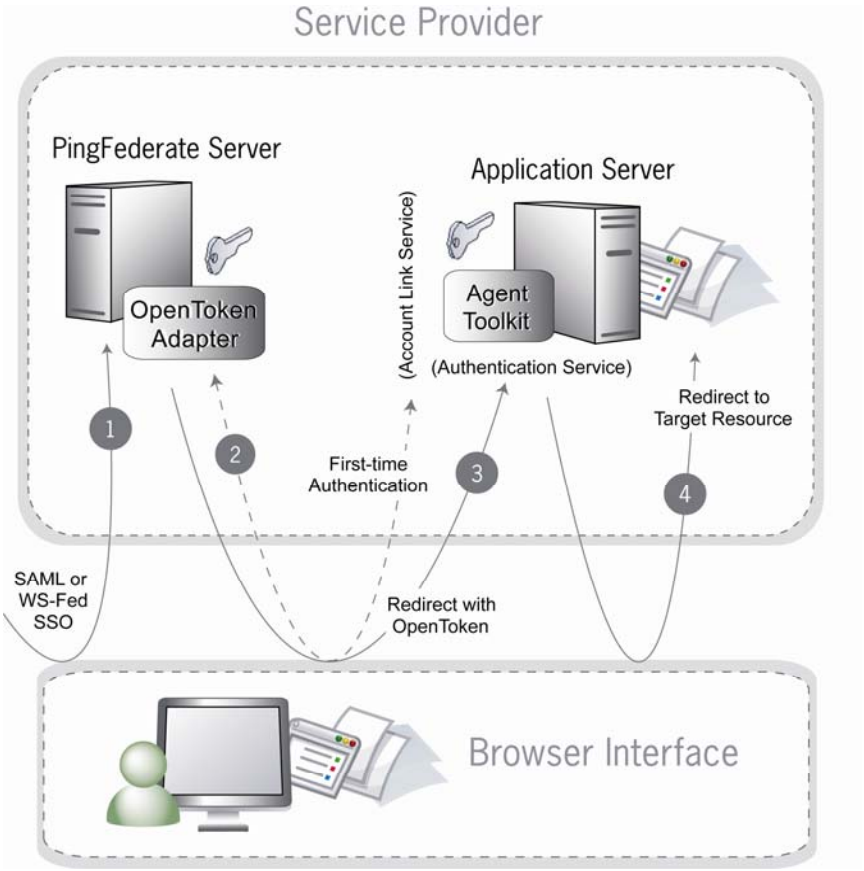
The Agent Toolkit for .NET receives multi-valued attributes passed in the SAML assertion from PingFederate as an `opentoken.MultiStringDictionary` collection of attributes. The following code snippet demonstrates how to get the multi-valued attributes in the `opentoken.MultiStringDictionary` collection using the Agent Toolkit:

```
try {
   //see "Using the Agent API" section for sample code
   //that instantiates and configures an Agent instance
```

```
    MultiStringDictionary userInfo =
    agent.ReadTokenMultiStringDictionary(Request);
    if(userInfo != null) {
       String username = userInfo[Agent.TOKEN_SUBJECT][0];
       List<String> groups = userInfo["GROUP"];
    }
}
catch(TokenException e) {
    // Handle exception
}
```

SP Single Sign-On (Using Account Linking)

If an SP's SSO implementation employs account linking, the flow of events is somewhat different since a user must authenticate to the SP application the first time SSO is initiated (for more information, see the "Key Concepts" chapter in the *Administrator's Manual*). In this case, PingFederate and the OpenToken Adapter support an integration mechanism to redirect the user to an Account Link Service to which a user can authenticate initially. Upon successful authentication, the user's browser is redirected back to PingFederate with an `OpenToken`, which PingFederate uses to create an account link for the user. For subsequent SSO requests, PingFederate uses the account link established in the first SSO request to identify the user. It then creates an `OpenToken` and sends it to the Authentication Service associated with the application.



Sequence

1. PingFederate receives an assertion under either the SAML 2.0 or WS-Federation protocol.
2. If this is the first time the user has initiated SSO to this SP, PingFederate redirects the browser to the Application Server's Account Link Service, where the user must authenticate. Upon successful authentication, an `OpenToken` is returned to PingFederate, and an account link is established for this user within PingFederate. This account link is used on subsequent SSO transactions.
3. PingFederate retrieves the local user ID from its account link data store. Through the OpenToken Adapter, PingFederate generates an `OpenToken` based on the assertion and account link. PingFederate then redirects the user's browser to the Web application's SSO Authentication Service, passing the `OpenToken` in the redirect.
4. The Authentication Service extracts the contents of the `OpenToken`, establishes a session for the user, and redirects the user's browser to the Target Resource (the `resumePath` URL sent as a query parameter).

In an Account Linking event, the user's browser is redirected to the configured Account Linking service in the SP OpenToken Adapter instance. The application should capture the `resumePath` upon a `GET` request to this URL with something similar to the following:

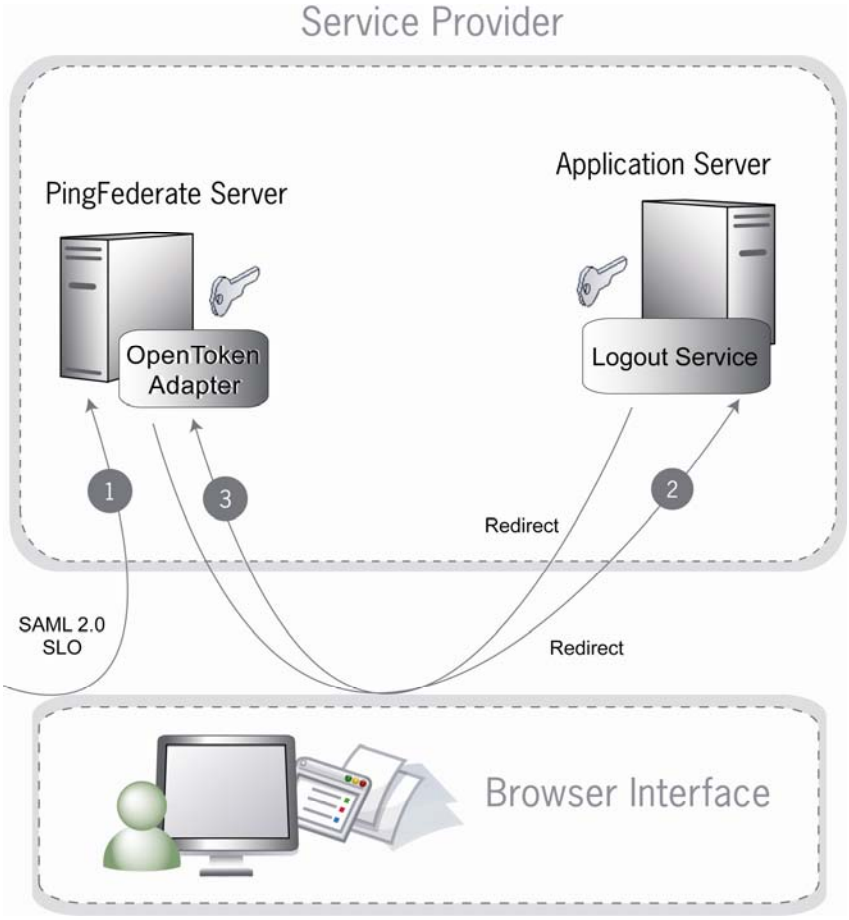
```
IDictionary userInfo = new Dictionary<String, String>();
// Add userId for the logged on user as the token subject
userInfo.Add(Agent.TOKEN_SUBJECT, <userId>);
String returnUrl = "https://<PingFederate DNS>:9031" + Request["resume"];
. . . .
try {
   UrlHelper urlHelper = new UrlHelper(returnUrl);
   //see "Using the Agent API" section for sample code
   //that instantiates and configures an Agent instance
   agent.WriteToken(userInfo,Response,urlHelper,false);
   returnUrl = urlHelper.ToString();
}
catch(TokenException e) {
   // Handle exception
}
Response.Redirect(returnUrl);
```

SP Single Logout (SLO)

When an SP PingFederate server receives a request for SLO, it redirects the user's browser to the Logout Service as configured in the SP OpenToken Adapter instance. As part of the redirect, PingFederate and the OpenToken Adapter include both an `OpenToken` and a `resumePath` query parameter.

- The `OpenToken` includes attributes about the user.
- The `resumePath` query parameter provides the target application URL.

A user can have multiple sessions. This logout sequence, as shown in the following diagram, will occur for each of the user's sessions controlled by the SP PingFederate server.



Sequence

1. PingFederate receives an SLO request under the SAML 2.0 protocol.
2. PingFederate, via the OpenToken Adapter, redirects the browser to the Application Server's Logout Service.
3. The Logout Service returns to PingFederate, indicating that the logout was successful.

The code needed to perform an SP SLO is identical to that required for an IdP SLO.

Testing

You can test the .NET Integration Kit using the sample application bundled with this distribution. (See the *.NET Sample Application Startup Guide* in the `/docs` directory.)