PingFederate®

PHP Integration Kit

Version 2.5.1

User Guide

PingIdentity®

PingFederate PHP Integration Kit *User Guide*
Version 2.5.1
December, 2012

Ping Identity Corporation
1001 17th Street, Suite 100
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

Document Lifetime

Ping Identity may occasionally update online documentation between releases of the related software. Consequently, if this PDF was not downloaded recently, it may not contain the most up-to-date information. Please refer to documentation.pingidentity.com for the most current information.

From the Web site, you may also download and refresh this PDF if it has been updated, as indicated by a change in this date:

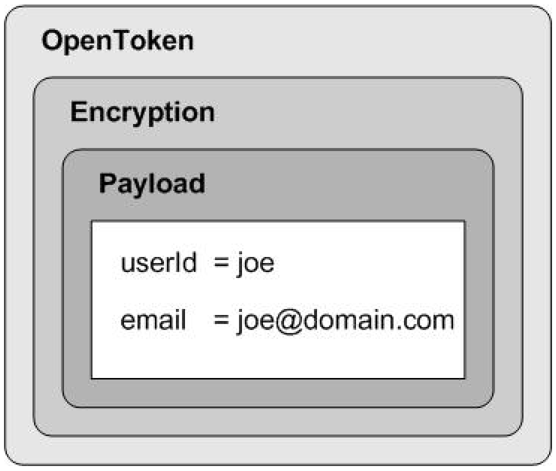
December 18, 2012

Contents

Introduction

The PHP Integration Kit includes the OpenToken Adapter and an Agent Toolkit for PHP, which allows developers to integrate their PHP 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 PHP IdP application. On the SP side, the kit allows a PHP SP application to receive user attributes from the SP server.

The PHP 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:



Intended Audience

This document is intended for PingFederate administrators and Web application developers who will customize one or more PHP 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 PHP Sample Application Startup Guide to have a working example.

System Requirements

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

- PingFederate 6.x (or higher)
- PHP 5.4.x (on the application Web server)
- `cURL` extension on the PHP server – enables IdP and SP applications to communicate with PingFederate

Note: This is required for both the Windows and Linux installations of PHP 5.4.8.

- `mcrypt` extension on the PHP server – used to encrypt and decrypt the `OpenToken`

  Find more information and the download at `http://php.net/mcrypt` and `http://mcrypt.sourceforge.net`.

- `mhash` extension on the PHP server – used to generate `hmacs` and secure keys from passwords

  Find more information and the download at `http://php.net/mhash` and `http://mhash.sourceforge.net.`

- `zlip` extension – used for data compression

  The PHP kit uses the `gzuncompress` method to uncompress data. For more information, see `http://php.net/manual/en/ref.zlib.php`.

ZIP Manifest

The distribution ZIP file for the PHP Integration Kit contains the following: #

- `ReadMeFirst.pdf` – contains links to this online documentation
- `/legal` – contains this document:
  - `Legal.pdf` – copyright and license information
- `/dist` - contains libraries needed to run the adapter:
  - `opentoken-adapter-2.5.1.jar` - OpenToken Adapter JAR file
  - `pingidentity/opentoken` – directory containing files for the Agent Toolkit for PHP
    - `agent.php`
    - `/helpers` – directory containing:

      – `config.php`

      – `keyvalueserializer.php`

      – `multistringarray.php`

      – `opentoken.php`

      – `passwordkeygenerator.php`

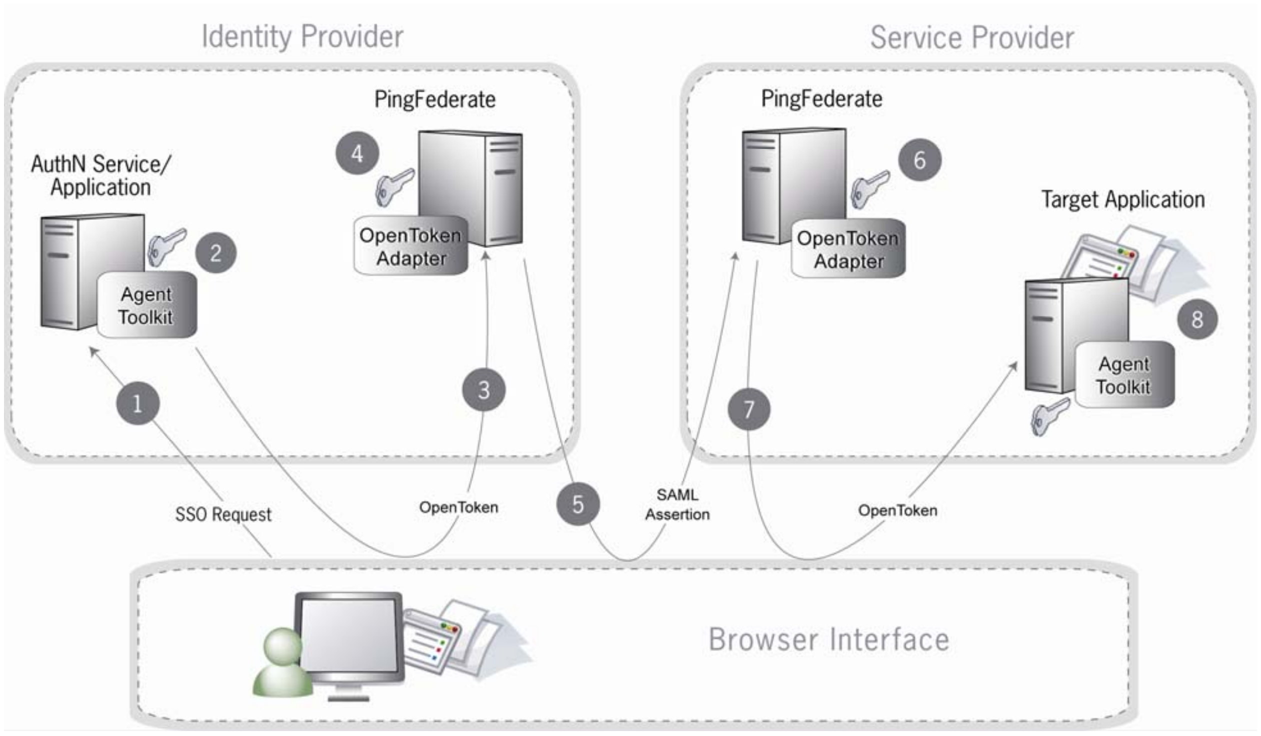
      – `token.php`
- `/sample`
  - `/common` – directory containing common files for the sample applications
  - `/config` – configuration directory for the sample applications
  - `/idp` – IdP Sample Application directory
  - `/sp` – SP Sample Application directory
  - `data.zip` – preconfigured PingFederate configuration archive for the sample applications
  - `install.sh` – installation shell script for the sample applications

Processing Overview

The PHP Integration Kit consists of two parts:

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

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



Processing Steps

1. A user initiates an SSO transaction.
2. The IdP application inserts user attributes into the Agent Toolkit for PHP, 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 PHP 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 Data Stores in the PingFederate *Administrator's Manual* for more information.

Installation and Setup

The following sections describe how to install and configure the OpenToken Adapter for both an IdP and an SP.

Installing the OpenToken Adapter and Configuring PingFederate

Note: If you have already deployed version 2.5.1 (or higher) of the OpenToken Adapter, skip steps 1 through 4 in the following procedure

1. Stop the PingFederate server if it is running.
2. Remove any existing OpenToken Adapter files (`opentoken*.jar`) from the directory:

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

   The adapter JAR file is `opentoken-adapter-<version>.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.

3. Copy `opentoken-adapter-2.5.1.jar` from the `/dist` directory to the PingFederate 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 OpenToken Adapter Configuration in the PingFederate *Administrator's Manual.*

   Note: Ensure **Obfuscate Password**, located in Advanced Fields is unchecked. The PHP agent does not support an encrypted password. The password will be Base64 encoded.

6. On the Actions screen, click the **Invoke Download** link and then click **Export** to download the `agent-config.txt` properties to a directory that is readable by the PHP server.

   Ensure that the file location is not part of the Web server's document root.

   Note: The `agent-config.txt` file *must not* be accessible via HTTP.

7. Once the adapter is configured, create a connection to your partner using that adapter instance. (For more information, see Identity Provider SSO Configuration or Service Provider SSO Configuration in the PingFederate *Administrator's Manual.*)
8. Ensure that `mcrypt, zlip` and `mhash` extensions are installed in the PHP server. Using the `phpinfo()` function, look for `mcrypt, zlip,` and `mhash` support in the resulting output.

9. Ensure that your PHP server is configured to process files with a `.php` extension and is using the appropriate version of PHP (see System Requirements on page 4).

   On the Apache HTTPD server, you can use the `AddType` command to add the extension:

   `AddType application/x-httpd-php .php` (not needed on the built-in PHP Web server)

10. Copy the `/dist/pingidentity/opentoken` directory into your PHP `include` path as `/opentoken`.

11. Edit the `AGENT_CONFIG_FILE` constant in `pingidentity/opentoken/helpers/config.php` to point to the location of the `agent-config.txt` file from Step 6.

    On Windows, be sure to use the network-path syntax:

    `\\<host>\<path>`

Note: To enforce UTF-8 encoding, explicitly specify `default_charset=utf8` in the PHP server `php.ini` file. If this is not set, PHP will use the default operating system character set, which can lead to internationalization compatibility issues.

Integrating with an IdP PingFederate Server

This section provides implementation guidelines and code examples for PHP 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 `Agent` API is a PHP object that provides access to functionality for writing an `OpenToken` to a given HTTP response.

The Agent API in PHP Kit 2.5.1 makes use of Namespaces, which enable the Agent API to be auto-loaded with other applications. For more information on auto-loading, see `http://php.net/manual/en/language.oop5.autoload.php`

Note: The sample code in this *User Guide* assumes that the Agent APIs are already loaded.

The following is sample code for auto-loading. Other auto-loading frameworks such as `Zend` (`http://framework.zend.com`) can be used instead.

```
spl_autoload_extensions(".php");
spl_autoload_register();
```

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

```
<?php
   use pingidentity\opentoken\agent;
   $myagent = new Agent();
?>
```

When the `Agent` object is instantiated, it uses the `config.php` file to find the configuration data generated when the OpenToken Adapter was configured. 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 file specified in `config.php` is not found, the agent constructor will throw an exception.

The `writeTokenToHTTPResponse` method takes an array of attributes and encodes them into an `OpenToken`, which is then written to the HTTP response.

Note: The array of attributes parameter *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 it out to the response, the *lastError* attribute of the agent instance will contain a message with a description of the error.

Sample Code

The code snippet below demonstrates the use of the `writeTokenToHTTPResponse` method:

```
<?php
   # Use and instantiate an agent
   use pingidentity\opentoken\agent;
   $myagent = new Agent();

   # Setup an array of values
   $myvalues = array(TOKEN_SUBJECT => "$userId");

   # Generate and write the token to a cookie (assuming that's
   # our configuration)
   $myagent->writeTokenToHTTPResponse($myvalues);
?>
```

Passing Multi-Value Attributes

The Agent Toolkit for PHP 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 `MultiStringArray` PHP class distributed with the Agent Toolkit for PHP.

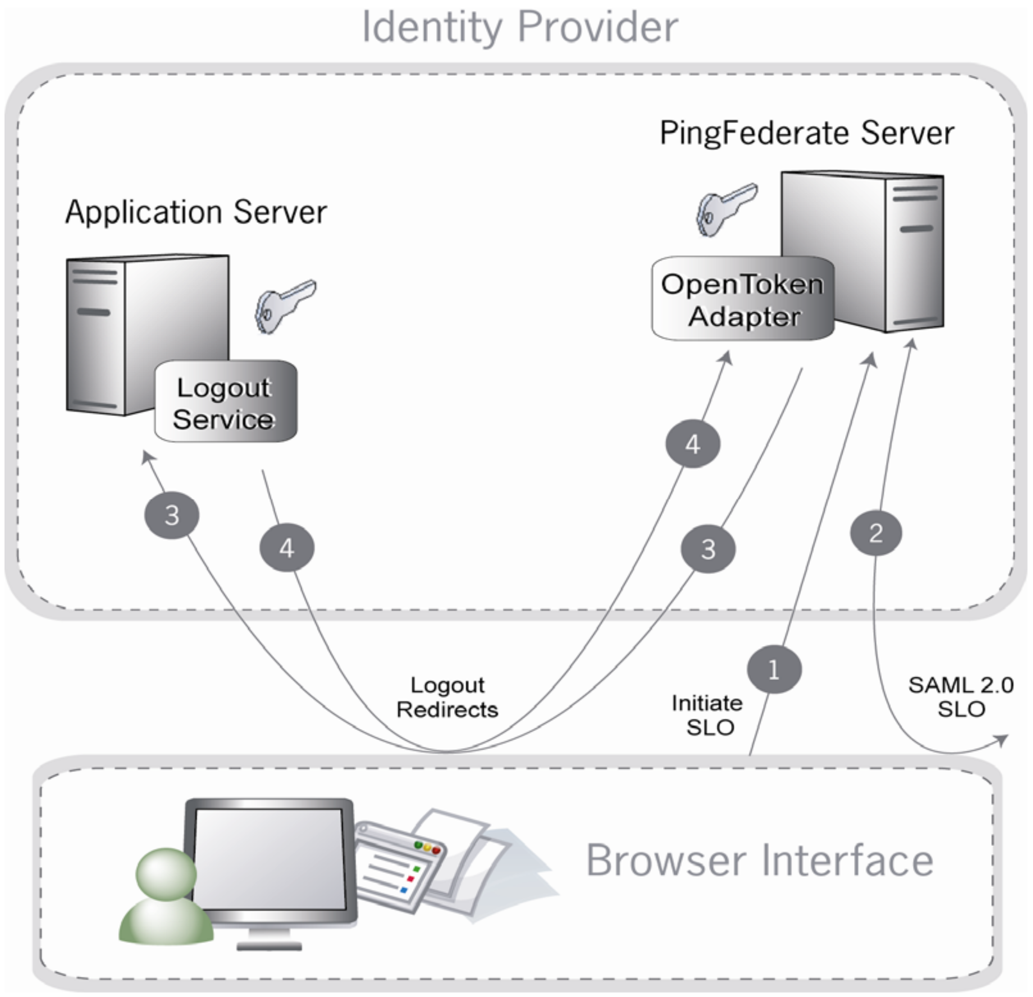
The following code snippet demonstrates how to pass multi-valued attributes using the Agent Toolkit for PHP:

```
<?php
   # Use and instantiate an agent
   use pingidentity\opentoken\agent;
   include_once "pingidentity/opentoken/helpers/multistringarray.php";
```

```
    $myagent = new Agent();

    # Setup an array of values
    $myvalues = MultiStringArray()

    # Single Value Attribute
    $myvalues->add(TOKEN_SUBJECT, <$userId>);

    #Multiple Value Attribute
    $myvalues->add(“MultiValueAttr”, “value1”);
    $myvalues->add(”MultiValueAttr”, ”value2”);
    $myvalues->add(“MultiValueAttr”, “value3”);

    # Generate and write the token to a cookie (assuming that's
    # our configuration)
    $myagent->writeTokenToHTTPResponse($myvalues);
?>
```

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



Processing Steps

1. User initiates a single logout request. The request targets the PingFederate server's `/idp/startSLO.ping` endpoint.
2. PingFederate sends a logout request and receives responses from 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.

Sample Code

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

```
<?php
   # Use and instantiate an agent
   use pingidentity\opentoken\agent;;
   $myagent = new Agent();
```

```
    # Setup an array of values
    $myvalues = array(TOKEN_SUBJECT => “user1”);

    # Generate and write the token to a string (assuming that's
    # our configuration)
    $queryParam = $myagent->writeTokenToHTTPResponse($myvalues);

    $resumePath = $_GET['resume'];
    $returnUrl = "https://<SERVER_NAME>:9031/" . $resumePath;

    session_destroy();
    ob_end_clean();

    if ($queryParam)
    {
      if (strpos($returnUrl, "?") == FALSE)
      {
        $returnUrl = $returnUrl . "?";
      }
      $returnUrl = $returnUrl . $queryParam;
    }

    header("Location: " . $returnUrl);
?>
```

Integrating with an SP PingFederate Server

This section provides implementation guidelines and code examples for PHP 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 included `OpenToken`. The application can use either the `Agent` object to do the decoding explicitly, or include the `opentoken.php` file to have that processing done automatically.

To do the processing automatically, include the `pingidentity/opentoken/helpers/opentoken.php` file inside the page responsible for logging the user in. The `opentoken.php` file will automatically parse inbound tokens using the agent configuration generated by PingFederate in Step 6 during the PHP Integration Kit installation process, and provide them as global variables.

These global variables are listed below:

Variable Name	Description/Contents
$opentoken_subject	String representing the authenticated subject in the `OpenToken`
$opentoken_haveValidToken	`Boolean` indicating whether the received `OpenToken` was valid
$opentoken_lastError	String containing the last error message from decoding the `OpenToken`
$opentoken_values	Keyed-array of values contained in the `OpenToken`. `Null` if no valid token was received

Sample Code

A simple example of this mode of operation is demonstrated in the PHP code below:

```
<?php
   include_once "pingidentity/opentoken/helpers/opentoken.php"
   if ($opentoken_haveValidToken == true)
   {
      echo "Welcome ", $opentoken_subject, "!<p>";
      print_r($opentoken_values);
   }
   else
   {
      echo "You are not logged in: ", $opentoken_lastError, "<p>";
   }
?>
```

As with the IdP, you can use the `pingidentity/opentoken/agent.php` API to read tokens directly. The `agent.php` API is a PHP object that provides access to functionality for reading an `OpenToken` from a given HTTP request.

Instantiating the agent object is done simply by using the PHP Agent Namespace and invoking the constructor, as in the example below:

```
<?php
   use pingidentity\opentoken\agent;;
   $myagent = new Agent();
?>
```

When the `Agent` object is instantiated, it uses the `config.php` file to find the configuration data generated when the OpenToken Adapter was configured. This configuration data includes the name of the cookie that the agent object will read. If the file specified in `config.php` is not found, the `Agent` constructor will throw an exception.

The `readTokenFromHTTPRequest` method inspects the cookie (or query parameters, depending on the configuration of the agent instance) and decodes the `OpenToken`, returning an array of attributes or `null` if no token is found or an error is encountered. In the case of an error, the `lastError` attribute will contain the error message.

The following code demonstrates the use of this method:

```
<?php
   # Include and instantiate an agent
   use pingidentity\opentoken\agent;";
   $myagent = new Agent();

   # Use the read method to extract the token values
   $myvalues = $myagent->readTokenFromHTTPRequest();
   if ($myvalues)
   {
      # Found some values - print each of them
      foreach ($myvalues as $key => $value)
      {
       echo $key . " = " . $value . "<br>";
      }
   }
   else
   {
      # No values - print the error message
      echo "Error reading token: " . $myagent->lastError;
   }
?>
```

Receiving Multi-Value Attributes

The Agent Toolkit for PHP receives multi-value attributes passed in the SAML assertion from PingFederate using the `MultiStringArray` PHP class distributed with the Agent Toolkit for PHP.

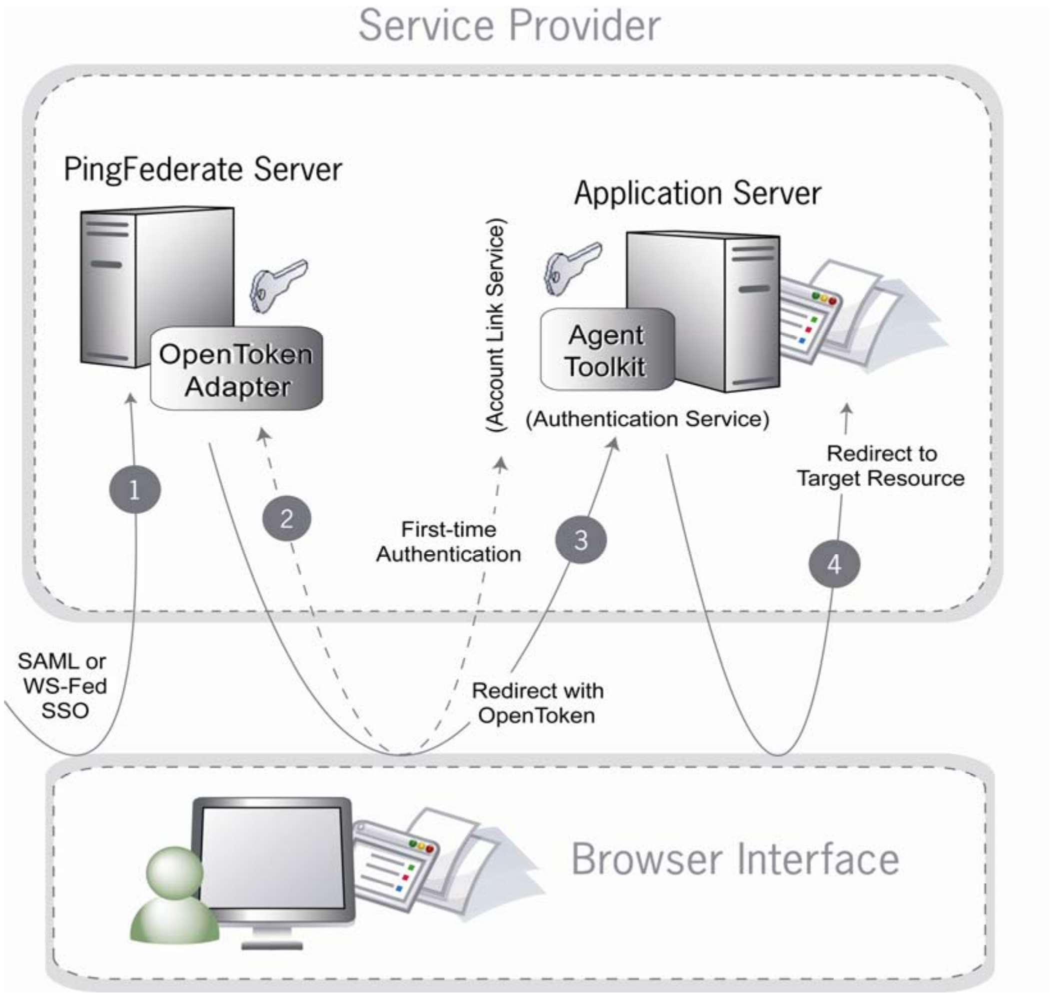
The following code snippet demonstrates how to get the multi-value attributes using the Agent Toolkit for PHP:

```
<?php
   # Include and instantiate an agent
   use pingidentity\opentoken\agent;
   use pingidentity\opentoken\helpers\multistringarray;

   $myagent = new Agent();

   # read OpenToken into a MultiStringArray ($userInfo)
   $userInfo = $myagent->readTokenFromHttpRequestToMultiStringArray();

   # retrieve single value attribute
   $username = $userinfo->get(TOKEN_SUBJECT, 0);
```

```
    # retrieve multiple value attribute.  Returns an array of values.
    $multiValues = $userinfo->get("MultiValueAttr");

?>
```

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 Key Concepts in the PingFederate *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 initially authenticates. Upon successful authentication, the account link service must redirect the user 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. PingFederate's OpenToken Adapter generates an `OpenToken` based on the assertion and account link, and 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 Link Service in the SP OpenToken Adapter instance.

Sample Code

The application should capture the `resumePath` upon a `GET` request to this URL with code similar to the following:

```
<?php
   # Include and instantiate an agent
   use pingidentity\opentoken\agent;
   $myagent = new Agent();

   $resumePath = $_GET['resume'];
   $returnUrl = "https://<SERVER_NAME>:9031/" . $resumePath;

   # After the user has authenticated, an OpenToken needs to be
   # generated with the userId and sent back to the previously
   # captured resumePath.

   $myvalues = array(TOKEN_SUBJECT => $userInfo["userid"],
   AUTHN_CTX_ATTRIBUTE_NAME => $userInfo[AUTHN_CTX_ATTRIBUTE_NAME]);

   $queryParam = $myagent->writeTokenToHTTPResponse($myvalues);
   if ($queryParam)
   {
       if (strpos($returnUrl, "?") == FALSE)
       {
          $returnUrl = $returnUrl . "?";
       }
       $returnUrl = $returnUrl . $queryParam;
   }
```

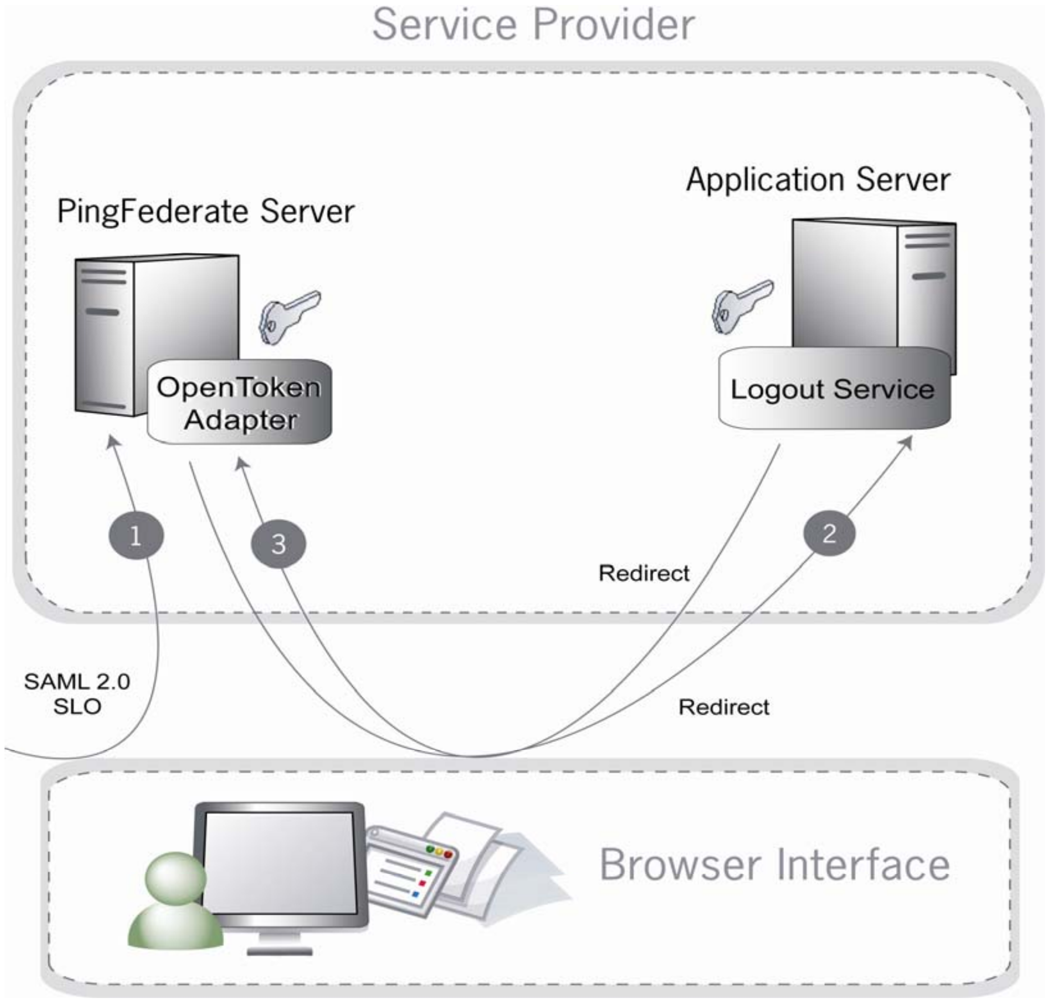
```
    header("Location: " . $returnUrl);
?>
```

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. (See Sample Code on page 11.)

Testing

You can test the PHP Integration Kit using the sample application bundled with this distribution. (See the PHP Sample Application Startup Guide.)