PingFederate®

OpenToken Token Translator
Version 1.0.1

User Guide

PingIdentity®

Version 1.0.1
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: http://www.pingidentity.com

Trademarks

Disclaimer

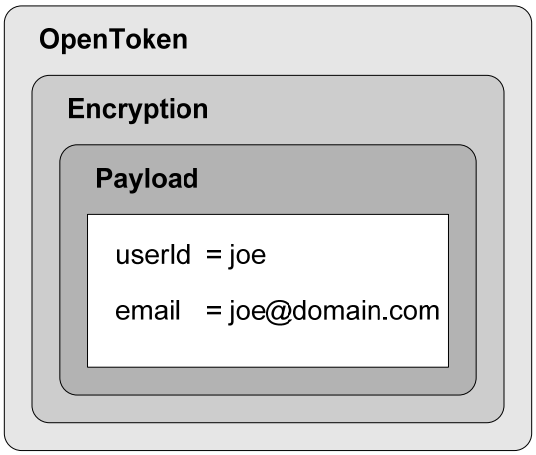
Contents

Introduction

The PingFederate OpenToken Token Translator provides a Token Processor and a Token Generator for use with the PingFederate WS-Trust Security Token Service (STS). The Token Processor allows an Identity Provider (IdP) STS to accept and validate an `OpenToken` from a Web Service Client (WSC) and then map user attributes into a SAML token for the WSC to send to a Web Service Provider (WSP). The Token Generator allows a Service Provider (SP) STS to issue an `OpenToken` for a WSP, including mapped attributes from an incoming SAML token.

> **Note**: Ping Identity provides a Java STS-Client Software Development Kit (SDK) for enabling Web Service applications (Client or Provider) to interact with the PingFederate STS. The SDK is available for download at www.pingidentity.com/support-and-downloads.

`OpenToken` is an open-standard, secure session cookie used to pass user information between an application and PingFederate. For STS purposes, the `OpenToken` is passed as a Web Services Security (WSS) binary security token in WS-Trust messages. The data within the `OpenToken` is a set of key/value pairs, encrypted using common encryption algorithms, as illustrated below:



This translator package includes a Java Application Programmer Interface (API) for WSC and WSP developers to use for writing or reading an `OpenToken`, respectively.

System Requirements

- PingFederate 6.0 or higher
- To use the Java API, J2SE Java Runtime Environment 1.5 or later is required on the WSC and WSP

  To use strong Advanced Encryption Standard (AES) encryption with a key size of more than 128 bits, the *Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy Files* must be installed in your JDK on PingFederate, as well as the WSC and WSP.

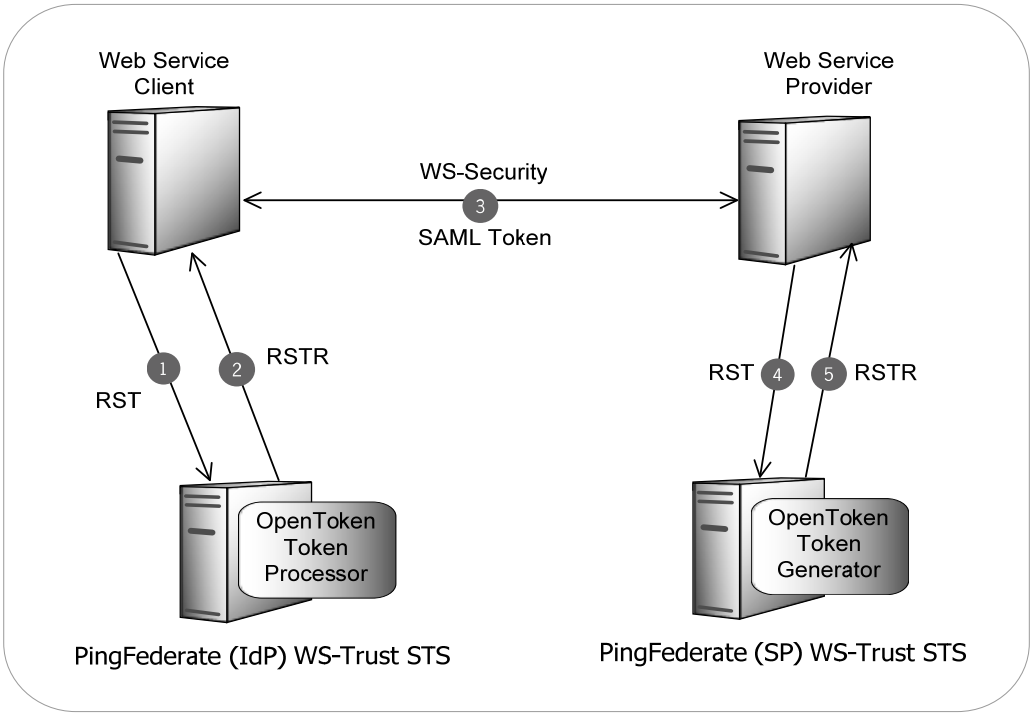
ZIP Manifest

The distribution ZIP file for the OpenToken Token Translator contains the following:

- `/docs` – contains documentation:
  - `OpenToken_STS_Token_Translator_Qualification_Statement.pdf` – testing and platform information
  - `OpenToken_STS_Token_Translator_User_Guide.pdf` – this document
- `/dist` – contains libraries needed to run the Token Translator and integrate with WSC and WSP applications:
  - `pf-opentoken-token-translator-1.0.1.jar` – the OpenToken Token Translator JAR file
  - `opentoken-agent-2.5.1.jar` – OpenToken Java Agent API
  - `commons-collections-3.2.jar` – Apache Commons Collections library
  - `commons-beanutils-1.7.0.jar` – Apache Commons Bean Utility library
  - `commons-logging-1.1.jar` – Apache Commons Logging library

WS-Trust STS Processing

The following figure shows a basic Web Services scenario using the PingFederate WS-Trust STS in the role of both IdP and SP:



Processing Steps

1. A WSC sends a Request Security Token (RST) message containing an `OpenToken` as a SOAP request to the PingFederate STS IdP endpoint.
2. The PingFederate `OpenToken` Token Processor validates the `OpenToken` and, if valid, maps attributes from the `OpenToken` into a SAML token. PingFederate issues the SAML token based upon the SP connection configuration and embeds the token in a Request Security Token Response (RSTR) which is returned to the WSC.
3. The WSC binds the issued SAML token into a WSS header and sends it via a SOAP request to the WSP.
4. The WSP sends an RST Issue request containing the SAML token to the PingFederate STS SP endpoint. PingFederate validates the SAML token and, if valid, maps attributes from the SAML token into an `OpenToken`. PingFederate issues the `OpenToken` based upon the OpenToken Token Generator configuration and embeds the token in an RSTR which is returned to the WSP.
5. The WSP receives the `OpenToken` in the RSTR for local domain processing.

IdP Installation and Setup

This section describes how to install and configure the OpenToken Token Processor for PingFederate acting as an IdP.

1. Copy the `pf-opentoken-token-translator-1.0.1.jar` file from the `dist` directory of this distribution to the `<pf-install>/pingfederate/server/default/deploy` directory of your PingFederate server installation.

   Note: If you have a previous version of the OpenToken Token Translator file installed, please delete it from the above location and replace it with the version referenced.

2. Log on to the PingFederate administrative console and click **Token Processors** under Application Integration Settings in the My IdP Configuration section of the Main Menu.

   If you do not see **Token Processors** on the Main Menu, enable WS-Trust by going to the Server Settings→Roles & Protocols screen and selecting WS-Trust for the IdP Role.

   Note: To enable token exchange, you may be prompted to provide SAML 1.x and SAML 2.0 federation identifiers for the STS on the Federation Info screen. Refer to the Federation Info screen's **Help** page for more information.

3. On the Manage Token Processor Instances screen, click **Create New Instance**.

4. On the Type screen, enter an Instance Name and Instance Id, and select OpenToken Token Processor 1.0.1 as the Type.

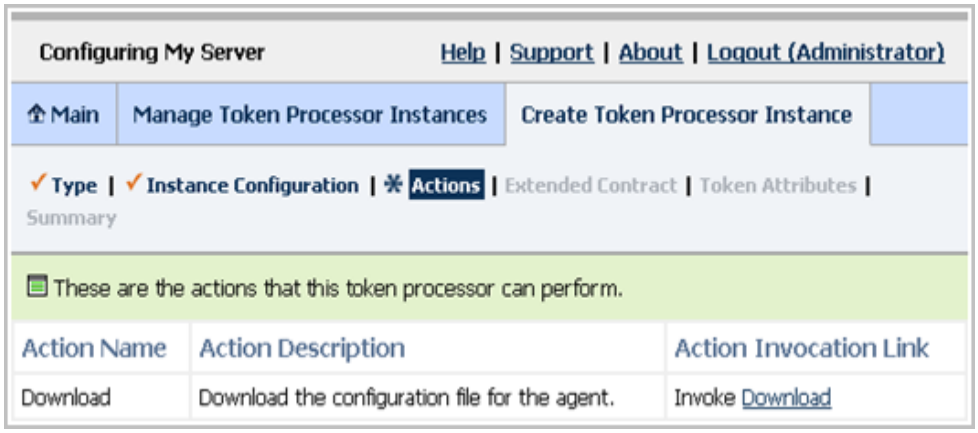
5. Click **Next**.

6. On the Instance Configuration screen, enter a strong password for generating the encryption key.

7. (Optional) Click **Show Advanced Fields** to set other encryption and validation options.

   Refer to the screen Description column for more information.

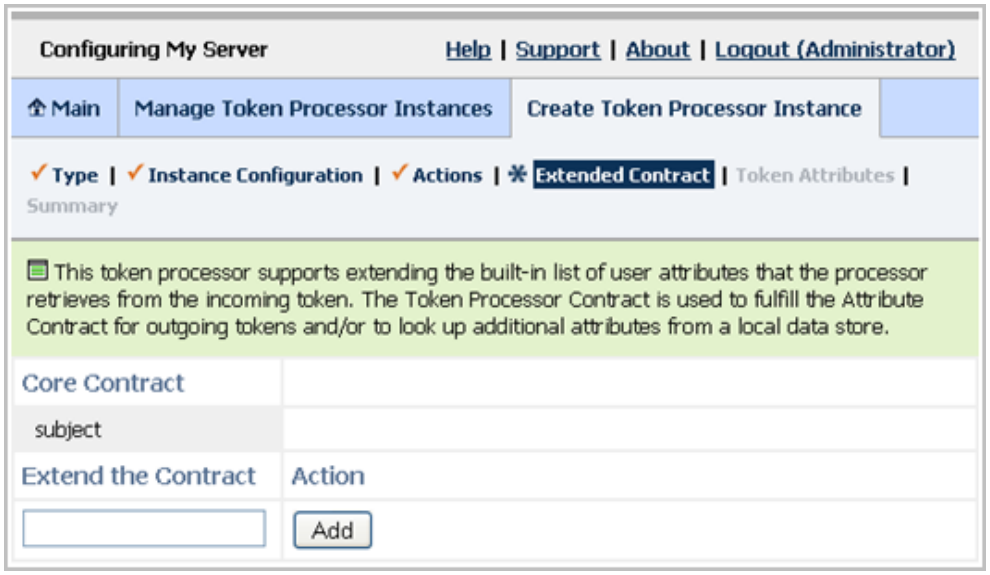
8. Click **Next**.



9. On the Actions page, click **Download** and then **Export** to save the `agent-config.txt` file.

   The WSC application that generates the `OpenToken` will need this information.

10. Click **Next**.



11. (Optional) On the Extended Contract screen, add any attributes that you want to map into the SAML assertion, in addition to the subject.

12. Click **Next**.

13. (Optional) On the Token Attributes screen, select any or all attributes whose values should be masked in PingFederate log files.

    Additionally, you may select **Mask all OGNL-expression generated log values**. (See the PingFederate *Administrator's Manual* for more information.)

14. Click **Next**.

15. On the Summary screen, verify that the information is correct and click **Done**.

16. On the Manage Token Processor Instances screen, click **Save**.

SP Installation and Setup

This section describes how to install and configure the OpenToken Token Generator for PingFederate acting as an SP.

1. Copy the pf-`opentoken-token-translator-1.0.1.jar` file from the `dist` directory of this distribution to the `<pf-install>/pingfederate/server/default/deploy` directory of your PingFederate server installation.

   Note: If you have a previous version of the OpenToken Token Translator file installed, please delete it from the above location and replace it with the version referenced.

2. Log on to the PingFederate administrative console and click **Token Generators** under Application Integration Settings in the My SP Configuration section of the Main Menu.

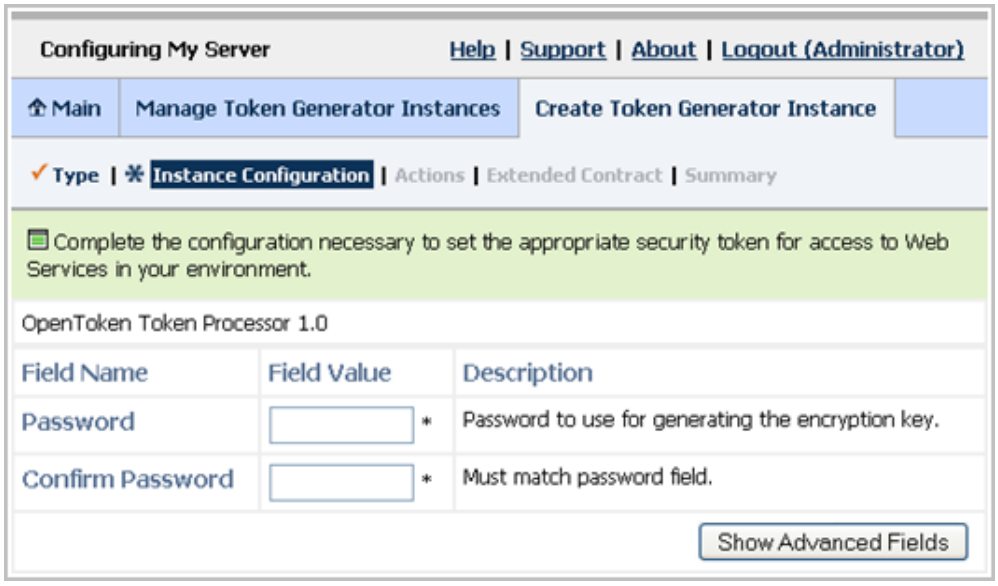
   If you do not see **Token Generators** on the Main Menu, enable WS-Trust by going to the Server Settings→Roles & Protocols screen and selecting WS-Trust for the SP Role.

   Note: To enable token validation, you may be prompted to provide SAML 1.x and SAML 2.0 federation identifiers for the STS on the Federation Info screen. Refer to the Federation Info screen's **Help** page for more information.

3. On the Manage Token Generator Instances screen, click **Create New Instance**.

4. On the Type screen, enter an Instance Name and Instance Id, and select OpenToken Token Generator 1.0.1 as the Type.

5. Click **Next**.



6. On the Instance Configuration page, enter in a strong password for generating the encryption key. Optionally, you can click **Show Advanced Fields** to set other encryption and validation options.

   Refer to the screen Description column for more information.

7. Click **Next**.

8. On the Actions page, click **Download** and then **Export** to save the `agent-config.txt` file. The WSP application that receives the `OpenToken` will need this information later.

9. Click **Next**.

10. (Optional) On the Extended Contract screen, add any attributes that you want to map from the SAML assertion, in addition to the subject.

11. Click **Next**.

12. On the Summary screen, verify that the information is correct and click **Done**.

13. On the Manage Token Generator Instances screen, click **Save**.

Using the OpenToken Agent API

WSC and WSP application developers can use the Java OpenToken Agent API (included in this distribution) to write and read an `OpenToken`. The API provide access to functionality for writing an `OpenToken` at the WSC to be exchanged for a SAML token, and for reading an `OpenToken` at the WSP that was issued from a SAML token.

Note: The Agent API may be used in conjunction with the Java STS Client SDK for interacting with the PingFederate WS-Trust endpoints. The SDK can be downloaded from pingidentity.com/products/downloads.cfm. Examples in this section implement the SDK.

Important: If you have already installed a previous version of the OpenToken Agent, we recommend removing the existing Agent JAR file and installing the current version to address potential security issues.

To install the Java OpenToken Agent API:

1. Copy the following JAR files to a location in the `CLASSPATH` of the Java application:
   - `opentoken-agent-2.5.1.jar`
   - `commons-collections-3.2.jar`

   Note: Version 3.2 or higher of the Apache Commons Collection is required.

2. Ensure the following Apache Commons libraries are also available in the application `CLASSPATH`:
   - `commons-beanutils-1.7.0.jar`
   - `commons-logging-1.1.jar`

Writing an OpenToken as a WSC

The OpenToken Agent API provides access to functionality for writing an `OpenToken` as a WSC to include in an Issue request to the PingFederate STS.

Java Sample Code

The `writeToken` method of the Agent class takes an `org.apache.commons.collections.MultiMap` collection of attributes and encodes them into an `OpenToken`.

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

If any errors are encountered while creating the token, a `TokenException` is thrown.

The code snippet below demonstrates generating an `OpenToken` and using the PingFederate STS Java Client SDK to send the `OpenToken` to the PingFederate STS:

```
// Configure the Opentoken agent
AgentConfiguration agentConfiguration = new AgentConfiguration();
agentConfiguration.setPassword("2Federate");
agentConfiguration.setCipherSuite(Token.CIPHER_SUITE_AES128CBC);

// Instantiate the OpenToken agent
Agent agent = new Agent(agentConfiguration);

// Set OpenToken attributes
MultiMap values = new MultiValueMap();
values.put(Agent.TOKEN_SUBJECT, "joe");
values.put("foo", "bar");
String tokenData = agent.writeToken(values);
```

```
// Configure STS Client
STSClientConfiguration idpStsConfig = new STSClientConfiguration();
idpStsConfig.setAppliesTo("http://sp.domain.com");
idpStsConfig.setStsEndpoint("https://idp.domain.com:9031/idp/sts.wst");
idpStsConfig.setInTokenType(TokenType.BINARY);
idpStsConfig.setInTokenValueType(TokenType.OPENTOKEN);

// Instantiate STS Client
STSClient idpStsClient = new STSClient(idpStsConfig);

// Send RST Issue request to STS
Element samlToken = idpStsClient.issueToken(tokenData);
```

Reading an OpenToken as a WSP

The Agent API provides access to functionality for reading an `OpenToken` received in an Issue request from the PingFederate STS.

Java Sample Code

The code snippet below demonstrates using the PingFederate STS Java Client SDK to retrieve the `OpenToken` issued from the PingFederate STS and using the OpenToken Agent API to read the `OpenToken`. If any errors are encountered while creating the token, a `TokenException` is thrown:

```
// Configure STS Client (SP-side, IdP connection)
AgentConfiguration spAgentConfiguration = new AgentConfiguration();
spAgentConfiguration.setPassword("Password1");
spAgentConfiguration.setCipherSuite(Token.CIPHER_SUITE_AES128CBC);

// Instantiate the OpenToken agent
Agent spAgent = new Agent(spAgentConfiguration);

// Configure STS Client
STSClientConfiguration spStsConfig = new STSClientConfiguration();
spStsConfig.setStsEndpoint("https://sp.domain.com:9031/sp/sts.wst");
spStsConfig.setOutTokenType(TokenType.BINARY);
spStsConfig.setOutTokenValueType(TokenType.OPENTOKEN);

// Instantiate STS Client
STSClient spStsClient = new STSClient(spStsConfig);

// Send RST Issue request to STS
Element opentoken = spStsClient.issueToken(new Saml20Token(samlToken));

// Read OpenToken
```

```
String otk = textContent(opentoken);
MultiMap otkValues = new MultiValueMap();
otkValues = spAgent.readTokenToMultiMap(otk);
```