**CitiFederation Developer Guide**

**Document**

**Version: 0.5**

**Written by**

**\*CATE CitiFederation**

**Document Revisions**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Date** | **Change details** | **Version** | **Modified By** | **Reviewed By** |
| 06/24/2014 | Initial Draft | 0.1 | Partha Peddi |  |
| 04/07/2017 | CitiSSO WS documentation | 0.2 | Partha Peddi |  |
| 04/20/2017 | Updated document content | 0.3 | Partha Peddi |  |
| 07/25/2017 | Added OpenToken with SAML IDP use case & custom login templates per application | 0.4 | Partha Peddi |  |
| 11/29/2017 | Added ChangePassword request parameters sample | 0.5 | Partha Peddi |  |
| 05/18/2018 | Added SessionValidation with user attribute retrieval use case | 0.6 | Partha Peddi |  |
| 06/20/2018 | Updated content for CitiSSO WS. OAuth2 content is update in progress. | 0.7 | Partha Peddi |  |
| 07/13/2018 | Updated OpenID Connect content | 0.7.1 | Partha Peddi |  |

**Contents**

[1 Introduction 4](#_Toc519273723)

[1.1 Assumptions 4](#_Toc519273724)

[1.2 Definitions 4](#_Toc519273725)

[1.3 Related Documents 4](#_Toc519273726)

[2 Client Best Practices 4](#_Toc519273727)

[2.1 Required 4](#_Toc519273728)

[2.2 Recommended 5](#_Toc519273729)

[3 CitiFederation use cases 6](#_Toc519273730)

[4 SAML WS-Trust STS use cases 7](#_Toc519273731)

[4.1 CitiFederation WS-Trust use cases are listed in the table below: 7](#_Toc519273732)

[4.2 CitiSSO Web Services use cases 7](#_Toc519273733)

[5 CitiFederation internal & external SAML Browser based use cases 8](#_Toc519273734)

[6 CitiFederation Services SAML (WS-Trust) Workflows 9](#_Toc519273735)

[6.1 SAML WS-Trust Workflows 9](#_Toc519273736)

[6.1.1 IDP use case 9](#_Toc519273737)

[6.1.2 SP use case 9](#_Toc519273738)

[7 CitiSSO Web Services (WS-Trust) Workflows 10](#_Toc519273739)

[7.1 User Authentication 10](#_Toc519273740)

[7.1.1 Positive User Authentication use case 10](#_Toc519273741)

[7.1.2 Negative User Authentication use case 10](#_Toc519273742)

[7.2 Session Validation 10](#_Toc519273743)

[7.2.1 Positive SessionValidation use case 10](#_Toc519273744)

[7.2.2 Negative Session Validation use case 10](#_Toc519273745)

[7.3 Session Validation with User Attribute Retrieval 10](#_Toc519273746)

[7.3.1 Positive Session Validation with User Attribute retrieval use case 10](#_Toc519273747)

[7.3.2 Negative Session Validation with User Attribute Retrieval use case 11](#_Toc519273748)

[7.4 Change Password Workflow 11](#_Toc519273749)

[7.4.1 Positive ChangePassword use case 11](#_Toc519273750)

[7.4.2 Negative ChangePassword use case 11](#_Toc519273751)

[8 Passing name/value pairs with OpenToken for SAML Browser based Workflows 12](#_Toc519273752)

[8.1 IDP use case – Passing Name/Value pairs with OpenToken Workflow where OpenToken contains SMSESSION 12](#_Toc519273753)

[8.1.1 OpenToken contains SMSESSION: 12](#_Toc519273754)

[8.1.2 IDP use case – Passing Name/Value pairs with OpenToken Workflow where OpenToken does not contain SMSESSION 13](#_Toc519273755)

[8.2 SP use case - Passing Name/Value pairs with OpenToken Workflow 15](#_Toc519273756)

[8.2.1 Application configured to receive OpenToken (in same domain or Cookie Provider is configured) 15](#_Toc519273757)

[8.2.2 Application configured with RedirectWithHeaders (in same domain or Cookie Provider configured) 15](#_Toc519273758)

[8.2.3 Application configured with RedirectWithSMS (not in same domain & Cookie Provider not configured) 16](#_Toc519273759)

[9 CitiFederation OAuth2 use cases 17](#_Toc519273760)

[9.1 Authorization Code Workflow 18](#_Toc519273761)

[9.1.1 Client requests and receives Authorization Code at Authorization Endpoint 18](#_Toc519273762)

[9.1.2 Client Requests Access Token at Token Endpoint 19](#_Toc519273763)

[9.1.3 Resource Server validates Access Token 20](#_Toc519273764)

[9.2 Implicit Workflow 21](#_Toc519273765)

[9.2.1 Client requests and receives Access Token at Authorization Endpoint 21](#_Toc519273766)

[9.2.2 Resource Server validates Access Token 22](#_Toc519273767)

[9.3 Client Credentials Workflow 24](#_Toc519273768)

[9.3.1 Client requests and receives Access Token at Authorization Endpoint 24](#_Toc519273769)

[9.3.2 Resource Server validates Access Token 25](#_Toc519273770)

[9.4 Resource Owner Credentials Workflow 27](#_Toc519273771)

[9.4.1 Client requests and receives Access Token at Authorization Endpoint 27](#_Toc519273772)

[9.4.2 Resource Server validates Access Token 28](#_Toc519273773)

[9.5 SAML 2.0 Bearer Workflow 30](#_Toc519273774)

[9.6 OpenID Connect Configuration Endpoint 31](#_Toc519273775)

[9.7 OpenID Connect Basic Client Workflow 32](#_Toc519273776)

[9.7.1 Client requests and receives Authorization Code at Authorization Endpoint 32](#_Toc519273777)

[9.7.2 Client Requests Access Token at Token Endpoint 33](#_Toc519273778)

[9.7.3 Resource Server validates Access Token 34](#_Toc519273779)

[9.7.4 Client validates ID token 35](#_Toc519273780)

[9.7.5 Get User Profile Attributes 35](#_Toc519273781)

[9.8 OpenID Connect 1.0 Implicit Client Profile Workflow 37](#_Toc519273782)

[9.8.1 Client requests and receives Access Token at Authorization Endpoint 37](#_Toc519273783)

[9.8.2 Resource Server validates Access Token 38](#_Toc519273784)

[9.8.3 Client or Resource Server validates ID token 39](#_Toc519273785)

[9.8.4 Get User Profile Attributes 40](#_Toc519273786)

[9.9 OAuth2 Token Refresh 41](#_Toc519273787)

[9.10 Grant Management – Revocation of Refresh Tokens 43](#_Toc519273788)

[9.10.1 Client revokes Refresh Token at Token Revocation Endpoint 43](#_Toc519273789)

[9.11 Grant Management by user 44](#_Toc519273790)

[9.12 Client/UserAgent client\_secret update with OTP from Ops 45](#_Toc519273791)

[10 Client Integration Requirements 46](#_Toc519273792)

[10.1 WS-Trust integration 46](#_Toc519273793)

[10.1.1 TrustStore & Client Authenticated SSL Setup 46](#_Toc519273794)

[10.1.2 Java based Clients 46](#_Toc519273795)

[10.2 OpenToken integration 46](#_Toc519273796)

[11 APPENDIX: SOAP Request-Response Samples 47](#_Toc519273797)

[11.1 IDP WS-Trust Samples 47](#_Toc519273798)

[11.2 User Authentication 51](#_Toc519273799)

[11.3 ChangePassword Sample: 53](#_Toc519273800)

[11.4 SP WS-Trust Samples 53](#_Toc519273801)

[12 APPENDIX: Java sample code for OpenToken 56](#_Toc519273802)

[13 APPENDIX: Java sample code for WS-Trust 59](#_Toc519273803)

# Introduction

This document details the CitiFederation Web Services (with WS-Trust), CitiSSO WS (with WS-Trust), ChangePassword service, and browser based OpenToken use cases for client integration. Sample HTTP client, SOAP request/response (RST/RSTR) samples and additional content is provided in the Appendix sections.

## Assumptions

* In the WS-Trust SOAP Request samples provided, the request variables should be replaced with valid values for the WS-Trust call to work.
* Certificate Authentication should be configured with CitiFederation for WS-Trust clients.

## Definitions

|  |  |
| --- | --- |
| **Name** | **Description** |
| SAML | Secure Assertion Markup Language |
| IDP App | For CitiFederation as IDP use case, the configured Citi application is the IDP App. |
| SP App | For CitiFederation as SP use case, the configured Citi application is the SP App. |
| CitiFederation | Federation infrastructure built with PingFederate product |
| OpenToken | A secure token format for transferring user attributes between an application and PingFederate server. |

## Related Documents

|  |  |
| --- | --- |
| **Name** | **Description** |
| <https://catecollaboration.citigroup.net/domains/security/idem_standards/aamstds/citifederation/standards/citiFederation-tsd.docx> | CitiFederation TSD |
| <https://catecollaboration.citigroup.net/domains/security/idem_standards/aamstds/citifederation/standards/VendorData/openToken/opentokenjars.zip> | OpenToken jar files |
| <https://catecollaboration.citigroup.net/domains/security/idem_standards/aamstds/citifederation/standards/VendorData/openToken/pf-dotnet-integration-kit.zip> | .NET OpenToken files |
| <https://www.pingidentity.com/content/developer/en/resources/openid-connect-developers-guide.html> | OpenID Connect Developers Guide |
| <https://www.pingidentity.com/content/developer/en/resources/oauth-2-0-developers-guide.html> | OAuth2 Developers Guide |

# Client Best Practices

## Required

* Request timeout must be set between 30 to 60 seconds. This setting will ensure processes don't remain tied up after 5 to 30 seconds of inactivity.
* DNS cache timeout must be set between 5 to 30 seconds. The Time-To-Live (TTL) timer will ensure that the DNS cache expires quickly and always stays fresh. Lowering the TTL value will enable rapid expiration and refreshing of DNS records, causing the new records to propagate faster across the world and avoid DNS spoofing attack.

java.security.Security.setProperty ("networkaddress.cache.ttl", "30"); // TTL set to 30 SECONDS

* The load balancer URL must always be used.
* New CitiSSO WS integrations are required to have a unique Agent & ACO names with the client application’s CSI App ID and the suffix “citissows-citifed” configured in CitiFederation & SiteMinder Policy Server configuration.

## Recommended

* Retry logic is required only for communication errors (SSL handshake issues/ request timeouts). Retry logic must be setup on request failure with a delay of 1 to 5 seconds between each request.
* Maximum number of retries on request failure can be set to 2.

# CitiFederation use cases

CitiFederation Service instance provides CitiSSO WS and SAML IDP & SP functionalities.

CitiFederation internal and external instances provides OAuth2, OIDC, and browser based SAML IDP & SP functionalities.

The following CitiFederation Service use cases are supported:

1. SAML WS – IDP use case (WS-Trust)
2. SAML WS – SP use case (WS-Trust)
3. CitiSSO WS - User authentication with CitiSSO credentials. (WS-Trust)
4. CitiSSO WS - Session validation with SMSESSION. (WS-Trust)
5. CitiSSO WS - Session validation with Attribute Retrieval (WS-Trust)
6. CitiSSO WS - ChangePassword services (RESTful service)

The following CitiFederation Internal & External OpenToken use cases are supported:

1. Browser based SAML IDP use case using OpenToken for name/value pairs
2. Browser based SAML SP use case using OpenToken for name/value pairs

The following CitiFederation OAuth2 & OIDC use cases are supported:

1. OAuth2 – Authorization Code
2. OAuth2 – Implicit
3. OAuth2 – Client Credentials
4. OAuth2 – Resource Owner
5. OAuth2 – SAML Bearer
6. OpenID Connect – Basic Client Profile
7. OpenID Connect – Implicit Client Profile

# SAML WS-Trust STS use cases

## CitiFederation WS-Trust use cases are listed in the table below:

|  |  |  |  |
| --- | --- | --- | --- |
| **WS-Trust Use Case** | **Input Token in RST** | **RSTR Contains** | **Description** |
| IDP | SMSESSION | SAML 2.0 Assertion | RST created with CitiSSO Session Token (SMSESSION) |
| IDP | CitiSSO user credentials | SAML 2.0 Assertion | RST created with CitiSSO user credentials |
| SP | SAML 2.0 Assertion | SMSESSION | RSTR contains CitiSSO Session Token (SMSESSION) |
| SP | SAML 2.0 Assertion | OpenToken | RSTR contains OpenToken value with user data as encrypted name/value pairs. |

## CitiSSO Web Services use cases

|  |  |  |  |
| --- | --- | --- | --- |
| **WS-Trust Use Case** | **Input Token in RST** | **RSTR Contains** | **Description** |
| User Authentication | CitiSSO user credentials | SMSESSION | RST created with CitiSSO user credentials. RSTR contains SMSESSION value. |
| User Authentication | CitiSSO user credentials | OpenToken | RST created with CitiSSO user credentials.  RSTR contains OpenToken with user data as encrypted name/value pairs. |
| User Authentication | CitiSSO user credentials | Success/Failure result | RST created with CitiSSO user credentials.  RSTR contains success or failure Boolean result. |
| Session Validation | SMSESSION | SMSESSION | RST created with existing SMSESSION value.  RSTR contains updated SMSESSION value. |
| Session Validation | SMSESSION | OpenToken | RST created with existing SMSESSION value.  RSTR contains OpenToken with encrypted name/value pairs used for attribute retrieval. |

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case** | **HTTP POST contails** | **Response Contains** | **Description** |
| ChangePassword per User Store | CitiSSO user credentials & new password  HTTP POST Fields (username, oldPassword, newPassword ) | SMSESSION value or the string literal “INVALID”. | CitiSSO user credentials and new password are HTTP-POST to ChangePassword service with a custom header “CitiFedCPHeader”. Response contains SMSESSION value, or the string “INVALID” in case of failure. |

# CitiFederation internal & external SAML Browser based use cases

OpenToken is a secure token format developed by PingIdentity as a secure means to transfer name/value pairs between applications. CitiFederation uses OpenToken for name/value pair transfer for browser based IDP and SP use cases.

The IDP clients generate OpenToken (per user) with user’s data. This OpenToken is consumed by CitiFederation to generate the requested output token. The user’s data included in the OpenToken is intended for processing by CitiFederation for including into the SAML 2.0 Assertion.

The SP clients consume the OpenToken (per user) with user’s data. These OpenTokens are generated by CitiFederation. RedirectWithHeaders and RedirectWithSMS use cases are also detailed.

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case** | **Input OpenToken** | **Output OpenToken** | **Description** |
| SAML IDP | User attributes + SMSESSION | NONE | Input OpenToken is created by IDP App and posted to IDP Initiated Federation URL. |
| SAML IDP | User attributes only | NONE | Input OpenToken is created by IDP App that does HTTP POST or OpenToken to resumePath instead of IDP initiated Federation URL. |
| SAML SP | NONE | OpenToken & resume URL | OpenToken & resume URL are posted to the configured SP App URL. |
| SAML SP | NONE | Assertion Attributes,  User LDAP Attributes | HTTP POST to the configured redirection URL at SP App. RedirectWithHeaders use case. |
| SAML SP | NONE | Assertion Attributes, User LDAP Attributes, SMSESSION | HTTP POST to the configured redirection URL (unauthenticated access required) at SP App. RedirectWithSMS use case. |

# CitiFederation Services SAML (WS-Trust) Workflows

## SAML WS-Trust Workflows

### IDP use case

#### Positive IDP use case

1. IDP application creates RST with user's SMSESSION cookie value or user credentials (base64 encoded username|password) as binary token.
2. IDP application connects to CitiFederation with client authenticated ssl (2-way ssl), and submits RST request to receive RSTR response containing SAML 2.0 Assertion.
3. IDP application extracts the SAML 2.0 Assertion from RSTR, and submits to its configured SP for user federation.

#### Negative IDP use case

1. IDP application creates RST with user's SMSESSION cookie value or user credentials (base64 encoded username|password) as binary token.
2. IDP application connects to CitiFederation with client authenticated ssl (2-way ssl), and submits request to receive RSTR response containing failure reason.
3. IDP application redirects user to error page or retry page depending on the reason for failure.

### SP use case

#### Positive SP use case

1. SP application receives SAML 2.0 Assertion for user federation.
2. SP application connects to CitiFederation with client authenticated ssl (2-way ssl), submits RST request to receive RSTR response containing SMSESSION as binary token.
3. SP application extracts the binary token (SMSESSION cookie value) from RSTR, sets it as the SMSESSION cookie value in the user's browser and redirects user to protected resource.

#### Negative SP use case

1. SP application receives SAML 2.0 Assertion for user federation.
2. SP application connects to CitiFederation with client authenticated ssl (2-way ssl), submits RST request to receive RSTR response containing failure reason.
3. SP application redirects user to error page or back to pre-configured IDP page depending on the reason for failure.

# CitiSSO Web Services (WS-Trust) Workflows

All applications integrating with CitiSSO WS WS-Trust use cases require the creation of a new Agent and ACO in SiteMinder configuration of CitiFederation infrastructure. CitiSSO Ops team will perform the required configuration changes in the background during integration. CitiFed Service URL will be unique for CitiSSO WS clients.

## User Authentication

### Positive User Authentication use case

1. Client application creates RST with user credentials (base64 encoded username|password) as binary token value.
2. Client application connects to CitiFederation with client authenticated ssl (2-way ssl), submits RST request to receive RSTR response containing SMSESSION value as binary token value. With an alternate configuration at CitiFederation Service instance, RSTR contains Boolean success response for the user authentication request.
3. Client application extracts the binary token (SMSESSION cookie value) from RSTR, sets it as the SMSESSION cookie value in the user's browser and redirects user to protected resource. Alternatively, Client application performs the action

### Negative User Authentication use case

1. Client application creates RST with user's SMSESSION cookie value or user credentials (base64 encoded username|password) as binary token value.
2. Client application connects to CitiFederation with client authenticated ssl (2-way ssl), submits RST request to receive RSTR response containing failure reason.
3. Client application redirects user to error page depending on the reason for failure.

## Session Validation

### Positive SessionValidation use case

1. Client application creates RST with user's SMSESSION cookie value as binary token value.
2. Client application connects to CitiFederation with client authenticated ssl (2-way ssl), submits RST request to receive RSTR response containing SMSESSION value as binary token value.
3. Client application extracts the binary token (SMSESSION cookie value) from RSTR, sets it as the SMSESSION cookie value in the user's browser and redirects user to protected resource.

### Negative Session Validation use case

1. Client application creates RST with user's SMSESSION cookie value as binary token value.
2. Client application connects to CitiFederation with client authenticated ssl (2-way ssl), submits RST request to receive RSTR response containing failure reason.
3. Client application redirects user to error page depending on the reason for failure.

## Session Validation with User Attribute Retrieval

### Positive Session Validation with User Attribute retrieval use case

1. Client application creates RST with user's SMSESSION cookie value as binary token value.
2. Client application connects to CitiFederation with client authenticated ssl (2-way ssl), submits RST request to receive RSTR response containing OpenToken value as binary token value that contains the configured user attributes in it. Note that the SMSESSION value is not present in the OpenToken value.
3. Client application extracts the binary token (OpenToken value) from RSTR, and extracts the user attributes from the OpenToken.

### Negative Session Validation with User Attribute Retrieval use case

1. Client application creates RST with user's SMSESSION cookie value as binary token value.
2. Client application connects to CitiFederation with client authenticated ssl (2-way ssl), submits RST request to receive RSTR response containing failure reason.
3. Client application redirects user to error page depending on the reason for failure.

## Change Password Workflow

### Positive ChangePassword use case

1. Client application creates a RESTful client and performs a HTTP POST of the fields: username, oldPassword and newPassword to the appropriate ChangePassword service URL using 2-way SSL for the connection.
2. The custom header “CitiFedCPHeader” with value “true” is submitted to the ChangePassword service URL for the requests to be processed.
3. CitiFederation ChangePassword service returns a successful response containing SMSESSION=<value>.
4. Client application extracts the SMSESSION cookie value from the response, sets it as the SMSESSION cookie value in the user's browser (to the application’s domain) and redirects user to protected resource.

### Negative ChangePassword use case

1. Client application creates a RESTful client and performs a HTTP POST of the fields: username, oldPassword and newPassword to the appropriate ChangePassword service URL using 2-way SSL for the connection.
2. Missing custom header “CitiFedCPHeader” (value = “true”) in the request to the ChangePassword service URL results in an error.
3. CitiFederation ChangePassword service returns a response containing a generic error string “INVALID” instead of the SMSESSION value.
4. Client application takes the necessary action to the error with change password.

# Passing name/value pairs with OpenToken for SAML Browser based Workflows

## IDP use case – Passing Name/Value pairs with OpenToken Workflow where OpenToken contains SMSESSION

### OpenToken contains SMSESSION:



***Figure:*** CitiFederation as IDP configured to receive custom name/value pairs with OpenToken. Here user is authenticated and has a valid SMSESSION cookie in the browser.

#### Positive use case

1. User authenticates to IDP App, and clicks the link to federate to SP.
2. IDP App generates the OpenToken containing name=citifedidpot, SMSESSION=<value>, “SUBJECT”=<uid>, and other name/value pairs.
3. IDP App performs auto-post of the OpenToken (name = citifeddipot) to IDP initiated federation URL.
4. CitiFederation validates the OpenToken value, the SMSESSION value, authorizes the user, and then generates the SAML 2.0 Assertion for auto-post to SP ACS URL.
5. User is federated to SP application.

#### Negative use case

1. User authenticates to IDP App, and clicks the link to federate to SP.
2. IDP App generates the OpenToken containing name=citifedidpot, SMSESSION=<value>, “SUBJECT”=<uid>, and other name/value pairs.
3. IDP App performs auto-post of the OpenToken (name = citifeddipot) to IDP initiated federation URL.
4. CitiFederation validates the OpenToken value, the SMSESSION value, authorizes the user.
5. In case OpenToken expires, it redirects user to error page configured within CitiFederation IDP Adapter.\
6. If SMSESSION is invalid/expired, CitiFederation redirects user to the configured Authentication URL and ignores the auto-post OpenToken prior to authentication.
7. If user authorization fails, user is redirected to the configured authorization error URL.

### IDP use case – Passing Name/Value pairs with OpenToken Workflow where OpenToken does not contain SMSESSION



***Figure:*** CitiFederation as IDP configured to receive custom name/value pairs with OpenToken. Here user is not authenticated or the SMSESSION cookie has expired due to idle timeout in the browser.

#### Positive use case

1. User clicks on IDP initiated Federation URL link (without valid SMSESSION in browser)
2. User is redirected to the IDP Authentication URL configured in the IDP Adapter (as Authentication URL) with a resumePath in QueryString.
3. User authenticates with SSO credentials at IDP Authentication URL.
4. IDP App generates the OpenToken containing name=citifedidpot, “SUBJECT”=<uid>, and other name/value pairs.
5. IDP App does an auto-post of the OpenToken value to the resumePath at the configured CitiFederation URL (hardcoded in IDP App per env).
6. CitiFederation validates the user’s SMSESSION value, then validates the OpenToken, then authorizes the user, then generates the SAML 2.0 Assertion for auto-post to SP ACS URL.

#### Negative use case

1. User clicks on IDP initiated Federation URL link (without valid SMSESSION in browser)
2. User is redirected to the IDP Authentication URL configured in the IDP Adapter (as Authentication URL) with a resumePath in QueryString.
3. User authenticates with SSO credentials at IDP Authentication URL.
4. IDP App generates the OpenToken containing name=citifedidpot, “SUBJECT”=<uid>, and other name/value pairs.
5. IDP App does an auto-post of the OpenToken value to the resumePath at the configured CitiFederation URL (hardcoded in IDP App per env).
6. CitiFederation validates the user’s SMSESSION value, then validates the OpenToken, then authorizes the user.
7. If SMSESSION is invalid, user is redirected to the configured error URL.
8. If OpenToken is invalid, user is redirected to the configured error URL.
9. If user is not authorized, user is redirected to the configured authorization error URL.

## SP use case - Passing Name/Value pairs with OpenToken Workflow

### Application configured to receive OpenToken (in same domain or Cookie Provider is configured)

#### Positive use case

1. IDP performs auto-post of SAML 2.0 Assertion to CitiFederation ACS URL.
2. CitiFederation SP Adapter configured with SP App URL in SP WAM Adapter as Authentication Service URL.
3. CitiFederation processes the SAML 2.0 Assertion, then authorizes the user, then generates the OpenToken for auto-post to the configured SP App URL.
4. The auto-post contains OpenToken with parameter name = citifedspot and resume URL with parameter name = resume

#### Negative use case

1. IDP performs auto-post of SAML 2.0 Assertion to CitiFederation ACS URL.
2. CitiFederation SP Adapter configured with SP App URL in SP WAM Adapter as Authentication Service URL.
3. CitiFederation processes the SAML 2.0 Assertion, then authorizes the user.
4. Invalid SAML 2.0 Assertion results in user redirection to the configured error URL.
5. User authorization failure results in user redirection to the configured authorization error URL.

### Application configured with RedirectWithHeaders (in same domain or Cookie Provider configured)

#### Positive use case

1. IDP performs auto-post of SAML 2.0 Assertion to CitiFederation ACS URL.
2. Either the SAML 2.0 Assertion contains AttributeStatement with special attribute values for TARGETURI & TARGETAPP, or these values are hardcoded within CitiFederation SP configuration.
3. CitiFederation SP Adapter is configured with CitiFederation RedirectWithHeaders URL as Authentication Service URL.
4. CitiFederation processes the SAML 2.0 Assertion, then authorizes the user, then sends OpenToken to RedirectWithHeaders for processing.
5. RedirectWithHeders processes the OpenToken’s name/value pairs and does auto-post to the configured TARGETAPP URL with name/value pairs configured to be posted to SP App.
6. SP App is either in the same domain as CitiFederation, or is configured with Cookie Provider to translate the SMSESSION to SP App domain.
7. In case of null values, the name/value pair is not posted in the auto-post to the TARGETAPP URL.

#### Negative use case

1. IDP performs auto-post of SAML 2.0 Assertion to CitiFederation ACS URL.
2. Either the SAML 2.0 Assertion contains AttributeStatement with special attribute values for TARGETURI & TARGETAPP, or these values are hardcoded within CitiFederation SP configuration.
3. CitiFederation SP Adapter is configured with CitiFederation RedirectWithHeaders URL as Authentication Service URL.
4. CitiFederation processes the SAML 2.0 Assertion, then authorizes the user, then sends OpenToken to RedirectWithHeaders for processing.
5. RedirectWithHeders processes the OpenToken’s name/value pairs and does auto-post to the configured TARGETAPP URL with name/value pairs configured to be posted to SP App.
6. If SAML 2.0 Assertion has validation errors, user is redirected to the configured error URL.
7. If TARGETAPP URL is missing in the SAML 2.0 Assertion, user is redirected to error URL.
8. In case of authorization failure, user is redirected to the authorization error URL.
9. If TARGETAPP URL fails the valid target domain check, user is redirected to error URL.

### Application configured with RedirectWithSMS (not in same domain & Cookie Provider not configured)

#### Positive use case

1. IDP performs auto-post of SAML 2.0 Assertion to CitiFederation ACS URL.
2. Either the SAML 2.0 Assertion contains AttributeStatement with special attribute values for TARGETURI & TARGETAPP, or these values are hardcoded within CitiFederation SP configuration.
3. CitiFederation SP Adapter is configured with CitiFederation RedirectWithSMS URL as Authentication Service URL.
4. CitiFederation processes the SAML 2.0 Assertion, then authorizes the user, then sends OpenToken to RedirectWithSMS URL for processing.
5. RedirectWithSMS processes the OpenToken’s name/value pairs and does auto-post to the configured TARGETAPP URL with SMSESSION value & name/value pairs configured to be posted to SP App.
6. For this use case, SP App is not same domain as CitiFederation, and is not configured with Cookie Provider.
7. On receiving the SMSESSION value at the TARGETAPP URL, it sets the SMSESSION to the user’s browser as the SP App’s domain cookie, and redirects to the TARGETURI URL (relative URL).
8. The SP App’s WebAgent processes the new SMSESSION value for protected resource access.

#### Negative use case

1. IDP performs auto-post of SAML 2.0 Assertion to CitiFederation ACS URL.
2. Either the SAML 2.0 Assertion contains AttributeStatement with special attribute values for TARGETURI & TARGETAPP, or these values are hardcoded within CitiFederation SP configuration.
3. CitiFederation SP Adapter is configured with CitiFederation RedirectWithSMS URL as Authentication Service URL.
4. CitiFederation processes the SAML 2.0 Assertion, then authorizes the user, then sends OpenToken to RedirectWithSMS URL for processing.
5. RedirectWithSMS processes the OpenToken’s name/value pairs, performs valid target domain check on the TARGETAPP URL.
6. If SAML 2.0 Assertion has validation errors, user is redirected to the configured error URL.
7. If TARGETAPP URL is missing in the SAML 2.0 Assertion, user is redirected to error URL.
8. In case of authorization failure, user is redirected to the authorization error URL.
9. If TARGETAPP URL fails the valid target domain check, user is redirected to error URL.

# CitiFederation OAuth2 use cases

CitiFederation internal and external instances support OAuth2 grant types for client integration. The following use cases are supported.

OAuth2 Grant Types

1. Authorization Code
2. Implicit
3. Client Credentials
4. Resource Owner Credentials
5. SAML 2.0 Bearer

OpenID Connect Profiles

1. OIDC 1.0 Basic Client Profile
2. OIDC 1.0 Implicit Client Profile

Additional resources

1. OAuth2 Token Refresh
2. Grant Management – Revocation of Refresh Tokens by Client or Refresh Server
3. Access Grant management using website
4. Client/UserAgent client\_secret update

## Authorization Code Workflow

Authorization Code workflow is used by web applications.

The workflow concentrates on the interactions between Client (user agent), Authorization Server, and Resource Server.



***Figure***: Authorization Code Grant Type workflow

### Client requests and receives Authorization Code at Authorization Endpoint

Client/UserAgent submits the request to the Authorization Endpoint of the Authorization Server (in Browser) to receive authorization code after successful user authentication followed by scope acceptance by user.

|  |  |  |
| --- | --- | --- |
| **Label** | **OAuth2 Parameter** | **Description** |
| Client ID | client\_id | Provided by SecOps |
| Client Authentication | client\_secret | Provided by SecOps |
| Allowed Grant Types | response\_type  grant\_type | response\_type=code is used for Authorization Code grant |
| Scope | scope | Scope of the request from client. This pre-defined scope is configured in the Authorization Server. User accepts the scope to give permission to the Client/UserAgent after successful authentication. |
| Code Challenge | code\_challenge | (optional) PKCE security |
| State | state | (optional) Value sent in this parameter is returned in the response to the Client without modification. |
| Redirect URI | redirect\_uri | Requires to be the redirect url configured in Authorization Server for the client. |

Sample Request:

|  |
| --- |
| POST /as/authorization.oauth2 HTTP/1.1  Content-Type: application/x-www-form-urlencoded  Host: secureaccess.sit.nam.citigroup.net  Content-Length: 58  Authorization: Basic Q0xJRU5UX0lEOkNMSUVOVF9TRUNSRVQ=  Connection: Keep-Alive  Cache-Control: no-cache  **POSTED Content:**  client\_id=<CLIENT\_ID>&client\_secret=\*\*\*\*\*\*\*\*\*\*\*\*\*\*&code\_challenge=\*\*\*\*\*&response\_type=code&scope=edit&state=\_\_\_\_ |

### Client Requests Access Token at Token Endpoint

Back channel request from Client to Token Endpoint of the Authorization server for exchanging the authorization code to get Access Token and Refresh Token.

|  |  |  |
| --- | --- | --- |
| **Label** | **OAuth2 Parameter** | **Description** |
| Client ID | client\_id |  |
| Client Authentication | client\_secret |  |
| Grant Type | authorization\_code |  |
| Authorization Code | code | Authorization Code received from the Authorization Server |
| Redirect URI | redirect\_uri | This value should be included if present in the response from the Authorization Server |
| Code Verifier | code\_verifier | Same value as code\_challenge provided in the initial request to the Authorization Endpoint |

Sample Request:

|  |
| --- |
| POST /as/token.oauth2 HTTP/1.1  Content-Type: application/x-www-form-urlencoded  Host: secureaccess.sit.nam.citigroup.net  Authorization: Basic Q0xJRU5UX0lEOkNMSUVOVF9TRUNSRVQ=  Content-Length: 248  charset=UTF-8  Connection: Keep-Alive  Cache-Control: no-cache  **POSTED Content:**  client\_id=<CLIENT\_ID>&grant\_type=authorization\_code&code=<AUTHORIZATION\_CODE>&response\_mode=&code\_verifier=\*\*\*\*\*\*\*\*\*&redirect\_uri=&access\_token\_manager\_id=&aud=  **Sample JSON server response:**  {"access\_token":"chrGDvfltj3dS413f8NpMp2T9QuQ","refresh\_token":"4krOXVmvYLMALk5QBzrHLISoXM5UbDZpY7urHW5hz4","token\_type":"Bearer","expires\_in":299} |

### Resource Server validates Access Token

Back channel request from Resource Server to Token Endpoint of the Authorization server for Access Token validation.

Sample Request:

|  |
| --- |
| POST /as/token.oauth2 HTTP/1.1  Content-Type: application/x-www-form-urlencoded  Host: secureaccess.sit.nam.citigroup.net  Authorization: Basic UkVTT1VSQ0VfU0VSVkVSX0NMSUVOVF9JRDpSRVNPVVJDRV9TRVJWRVJfQ0xJRU5UX1NFQ1JFVA==  Content-Length: 215  charset=UTF-8  Connection: Keep-Alive  Cache-Control: no-cache  **POSTED Content:**  grant\_type=urn:pingidentity.com:oauth2:grant\_type:validate\_bearer&token=\*\*\*\*\*\*\*\*\*  **Sample JSON server response:**  {"access\_token":{"UserName":"pp51119","OrgName":"Citigroup Inc."},"scope":"","token\_type":"urn:pingidentity.com:oauth2:validated\_token","expires\_in":294,"client\_id":"rs\_client"} |

## Implicit Workflow

Implicit Workflow is used by mobile and desktop thick client use cases.

With desktop thick client, the Client/UserAgent and Resource Server are part of the thick client. The thick client implementation creates the HTTP Client for handling connection to the authorization server for user authentication and scope acceptance.

With the mobile application use case, the app creates an instance of WebView with the html form post to the authorization server for user authentication and scope acceptance.

Since the Implicit grant type is inherently not as secure as Authorization Code grant type, Refresh Token is not returned in the response from the Authorization Endpoint.

The workflow details the interactions between Client (user agent) & Resource Server with CitiFed as Authorization Server.



***Figure***: Implicit Grant Type workflow

### Client requests and receives Access Token at Authorization Endpoint

Client/UserAgent submits the request to the Authorization Endpoint of the Authorization Server to receive AccessToken after successful user authentication followed by scope acceptance by user.

|  |  |  |
| --- | --- | --- |
| **Label** | **OAuth2 Parameter** | **Description** |
| Client ID | client\_id | Provided by SecOps |
| Client Authentication | client\_secret | Provided by SecOps |
| Allowed Grant Type | response\_type | response\_type=token is used for Implicit grant |
| Scope | scope | Scope of the request from client. This pre-defined scope is configured in the Authorization Server. User accepts the scope to give permission to the Client/UserAgent after successful authentication. |
| State | state | (optional) Value sent in this parameter is returned in the response to the Client without modification. |
| Redirect URI | redirect\_uri | Requires to be the redirect url configured in Authorization Server for the client. |

Sample Request:

|  |
| --- |
| POST /as/authorization.oauth2 HTTP/1.1  Content-Type: application/x-www-form-urlencoded  Host: secureaccess.sit.nam.citigroup.net  Content-Length: 134  Connection: Keep-Alive  Cache-Control: no-cache  **POSTED Content:**  client\_id=<CLIENT\_ID>&client\_secret=\*\*\*\*\*\*\*\*\*\*\*\*\*\*&response\_type=token&scope=edit&state=\_\_\_\_  Server Response Sample:  {"access\_token":"chrGDvfltj3dS413f8NpMp2T9QuQ","token\_type":"Bearer","expires\_in":299} |

### Resource Server validates Access Token

Back channel request from Resource Server to Token Endpoint of the Authorization server for Access Token validation.

Sample Request:

|  |
| --- |
| POST /as/token.oauth2 HTTP/1.1  Content-Type: application/x-www-form-urlencoded  Host: secureaccess.sit.nam.citigroup.net  Authorization: Basic UkVTT1VSQ0VfU0VSVkVSX0NMSUVOVF9JRDpSRVNPVVJDRV9TRVJWRVJfQ0xJRU5UX1NFQ1JFVA==  Content-Length: 215  charset=UTF-8  Connection: Keep-Alive  Cache-Control: no-cache  **POSTED Content:**  grant\_type=urn:pingidentity.com:oauth2:grant\_type:validate\_bearer&token=\*\*\*\*\*\*\*\*\*  **Sample JSON server response:**  {"access\_token":{"UserName":"pp51119","OrgName":"Citigroup Inc."},"scope":"","token\_type":"urn:pingidentity.com:oauth2:validated\_token","expires\_in":294,"client\_id":"rs\_client"} |

## Client Credentials Workflow

Client Credentials Workflow is used for legacy OAuth2 client implementations where the user credentials are passed to the Client/UserAgent to obtain the Access Token, or when an interactive Client/UserAgent is not available.

The workflow details the interactions between Client (user agent) & Resource Server with CitiFed as Authorization Server’s Token Endpoint.



***Figure***: Resource Owner Credentials Grant Type workflow

### Client requests and receives Access Token at Authorization Endpoint

Client/UserAgent submits the request to the Authorization Endpoint of the Authorization Server to receive AccessToken after successful user authentication followed by scope acceptance by user.

|  |  |  |
| --- | --- | --- |
| **Label** | **OAuth2 Parameter** | **Description** |
| Client ID | client\_id | Provided by SecOps |
| Client Secret | client\_secret | Provided by SecOps |
| Allowed Grant Type | grant\_type | grant\_type=client\_credentials is used for Client Credentials grant |
| Scope | scope | Scope of the request from client. This pre-defined scope is configured in the Authorization Server. User accepts the scope to give permission to the Client/UserAgent after successful authentication. |
| State | state | (optional) Value sent in this parameter is returned in the response to the Client without modification. |
| Redirect URI | redirect\_uri | Requires to be the redirect url configured in Authorization Server for the client. |

Sample Request:

|  |
| --- |
| POST /as/token.oauth2 HTTP/1.1  Host: gmwgtdcseng02l.nam.nsroot.net:9031  Content-Type: application/x-www-form-urlencoded  Connection: Keep-Alive  Cache-Control: no-cache  Authorization: Basic Y2NfY2xpZW50OlQ3NnQ4N2xkc3Uxd2V4S0x6dUR1Q2pYUndZTWpzZkQyTXE1SlBEMHA0UWxzelRjbzgwbm9CQ1ROenZZTTFmMUc=  **POSTED Content:**  client\_id=<CLIENT\_ID>&grant\_type=client\_credentials  Server Response Sample:  {  "access\_token": "Yieb6GftzySDasQyBbubVwqLAwrS",  "token\_type": "Bearer",  "expires\_in": 299  } |

### Resource Server validates Access Token

Back channel request from Resource Server to Token Endpoint of the Authorization server for Access Token validation.

Sample Request:

|  |
| --- |
| POST /as/token.oauth2 HTTP/1.1  Content-Type: application/x-www-form-urlencoded  Host: secureaccess.sit.nam.citigroup.net  Authorization: Basic UkVTT1VSQ0VfU0VSVkVSX0NMSUVOVF9JRDpSRVNPVVJDRV9TRVJWRVJfQ0xJRU5UX1NFQ1JFVA==  Content-Length: 215  charset=UTF-8  Connection: Keep-Alive  Cache-Control: no-cache  **POSTED Content:**  grant\_type=urn:pingidentity.com:oauth2:grant\_type:validate\_bearer&token=\*\*\*\*\*\*\*\*\*  **Sample JSON server response:**  {"access\_token":{"UserName":"pp51119","OrgName":"Citigroup Inc."},"scope":"","token\_type":"urn:pingidentity.com:oauth2:validated\_token","expires\_in":294,"client\_id":"rs\_client"} |

## Resource Owner Credentials Workflow

Resource Owner Credentials Workflow is used for legacy OAuth2 client implementations where the functional user (FID) credentials are passed to the Client/UserAgent to obtain the Access Token, or when an interactive Client/UserAgent is not available.

The workflow details the interactions between Client (user agent) & Resource Server with CitiFed as Authorization Server’s Token Endpoint.



***Figure***: Resource Owner Credentials Grant Type workflow

### Client requests and receives Access Token at Authorization Endpoint

Client/UserAgent submits the request to the Authorization Endpoint of the Authorization Server to receive AccessToken after successful user authentication followed by scope acceptance by user.

|  |  |  |
| --- | --- | --- |
| **Label** | **OAuth2 Parameter** | **Description** |
| Client ID | client\_id | Provided by SecOps |
| Client Authentication | client\_secret | Provided by SecOps |
| Allowed Grant Type | response\_type | response\_type=token is used for Implicit grant |
| Scope | scope | Scope of the request from client. This pre-defined scope is configured in the Authorization Server. User accepts the scope to give permission to the Client/UserAgent after successful authentication. |
| State | state | (optional) Value sent in this parameter is returned in the response to the Client without modification. |
| Redirect URI | redirect\_uri | Requires to be the redirect url configured in Authorization Server for the client. |
| UserName | username | FID username |
| Password | password | FID password |

Sample Request:

|  |
| --- |
| POST /as/authorization.oauth2 HTTP/1.1  Content-Type: application/x-www-form-urlencoded  Host: secureaccess.sit.nam.citigroup.net  Authorization: Basic PFJPX0NMSUVOVF9JRD46PFJPX0NMSUVOVF9QQVNTV09SRD4=  Content-Length: 134  Connection: Keep-Alive  Cache-Control: no-cache  **POSTED Content:**  grant\_type=password&username=citimfaFID&password=\*\*\*\*\*\*\*\*\*\*\*\*\*\*&scope=edit&state=\_\_\_\_  Server Response Sample:  {"access\_token":"chrGDvfltj3dS413f8NpMp2T9QuQ","token\_type":"Bearer","expires\_in":299} |

### Resource Server validates Access Token

Back channel request from Resource Server to Token Endpoint of the Authorization server for Access Token validation.

Sample Request:

|  |
| --- |
| POST /as/token.oauth2 HTTP/1.1  Content-Type: application/x-www-form-urlencoded  Host: secureaccess.sit.nam.citigroup.net  Authorization: Basic UkVTT1VSQ0VfU0VSVkVSX0NMSUVOVF9JRDpSRVNPVVJDRV9TRVJWRVJfQ0xJRU5UX1NFQ1JFVA==  Content-Length: 215  charset=UTF-8  Connection: Keep-Alive  Cache-Control: no-cache  **POSTED Content:**  grant\_type=urn:pingidentity.com:oauth2:grant\_type:validate\_bearer&token=\*\*\*\*\*\*\*\*\*  **Sample JSON server response:**  {"access\_token":{"UserName":"pp51119","OrgName":"Citigroup Inc."},"scope":"","token\_type":"urn:pingidentity.com:oauth2:validated\_token","expires\_in":294,"client\_id":"rs\_client"} |

## SAML 2.0 Bearer Workflow

TBD

## OpenID Connect Configuration Endpoint

The public CitiFed OpenID Connect Configuration Endpoint provides clients with configuration information to use OpenID Connect protocol.

Basic Client workflow uses the OAuth2 Authorization Code grant type workflow, but includes OpenID scope in the token returned by the Authorization Server.

|  |  |
| --- | --- |
| **Label** | **Description** |
| Grant Management URL | <https://secureaccess.nam.citigroup.net/.well-known/openid-configuration> |

## OpenID Connect Basic Client Workflow

Basic Client workflow uses the OAuth2 Authorization Code grant type workflow, but includes OpenID scope in the token returned by the Authorization Server.



**Figure:** OpenID Connect 1.0 Basic Client Profile workflow

### Client requests and receives Authorization Code at Authorization Endpoint

Client/UserAgent submits the request to the Authorization Endpoint of the Authorization Server (in Browser) to receive authorization code after successful user authentication followed by scope acceptance by user.

|  |  |  |
| --- | --- | --- |
| **Label** | **OAuth2 Parameter** | **Description** |
| Client ID | client\_id | Provided by SecOps |
| Client Authentication | client\_secret | Provided by SecOps |
| Allowed Grant Types | response\_type  grant\_type | response\_type=code is used for Authorization Code grant |
| Scope | scope | scope=openid for OIDC Basic Client Profile. Additional scopes can be added here. |
| Code Challenge | code\_challenge | (optional) PKCE security |
| State | state | (optional) Value sent in this parameter is returned in the response to the Client without modification. |
| Redirect URI | redirect\_uri | Requires to be the redirect url configured in Authorization Server for the client. |

Sample Request:

|  |
| --- |
| POST /as/authorization.oauth2 HTTP/1.1  Content-Type: application/x-www-form-urlencoded  Host: secureaccess.sit.nam.citigroup.net  Content-Length: 58  Authorization: Basic Q0xJRU5UX0lEOkNMSUVOVF9TRUNSRVQ=  Connection: Keep-Alive  Cache-Control: no-cache  **POSTED Content:**  client\_id=<CLIENT\_ID>&client\_secret=\*\*\*\*\*\*\*\*\*\*\*\*\*\*&code\_challenge=\*\*\*\*\*&response\_type=code&scope=openid&state=\_\_\_\_ |

### Client Requests Access Token at Token Endpoint

Back channel request from Client to Token Endpoint of the Authorization server for exchanging the authorization code to get Access Token and Refresh Token.

|  |  |  |
| --- | --- | --- |
| **Label** | **OAuth2 Parameter** | **Description** |
| Client ID | client\_id |  |
| Client Authentication | client\_secret |  |
| Grant Type | authorization\_code |  |
| Authorization Code | code | Authorization Code received from the Authorization Server |
| Code Verifier | code\_verifier | Same value as code\_challenge provided in the initial request to the Authorization Endpoint |

Sample Request:

|  |
| --- |
| POST /as/token.oauth2 HTTP/1.1  Content-Type: application/x-www-form-urlencoded  Host: secureaccess.sit.nam.citigroup.net  Authorization: Basic Q0xJRU5UX0lEOkNMSUVOVF9TRUNSRVQ=  Content-Length: 248  charset=UTF-8  Connection: Keep-Alive  Cache-Control: no-cache  **POSTED Content:**  client\_id=<CLIENT\_ID>&grant\_type=authorization\_code&code=<AUTHORIZATION\_CODE>&response\_mode=&code\_verifier=\*\*\*\*\*\*\*\*\*&redirect\_uri=&access\_token\_manager\_id=&aud=  **Sample JSON server response:**  {"access\_token":"eyJhbGciOiJSUzI1NiIsImtpZCI6ImNpdGlmZWRfa2V5In0.eyJzY29wZSI6WyJvcGVuaWQiXSwiY2xpZW50X2lkIjoiYWNfb2ljX2NsaWVudCIsImlzcyI6InBmODMiLCJhdWQiOiJwcDUxMTE5Iiwic3ViIjoicHA1MTExOSIsImV4cCI6MTUzMTUxMjIxN30.jn3V-VjOaneF7yWZ1z0VvXEeH57ACeApAQNkxWqaJX8Mrqk\_uEAPlcP1y1NOzePkkDm-zuJsEKbHTiizdKWttJcbwjY29WWX5aqMY6qONhFFTsD4IgW1uEqcLmZVVTFC3ajILGH71hguvyWlrTj5AJtPadZAW8KM8cAgckzKTZ98zrfpA-8uwGratuqd1PzOTDZcRKqt--MDwd2KEb52rZiUA24GFyBLqh-bJ7xzXrAHDUaOgO4Y\_1hy\_l-lg2-L1jGCiTaj-B73f34yGtuwSXfXBne9PJgeJrZsWg7bqOkFahTh6csZpMFKG4QP5rrKCuMB0yKmQBvPnSrCnZdgCQ","refresh\_token":"qh9SJAfzC4coo2Yr0oZvneo1zfCgY7yDcIHzFHezAS","id\_token":"eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJwcDUxMTE5IiwiYXVkIjoiYWNfb2ljX2NsaWVudCIsImp0aSI6Ik1Tc01ZTzM1WXBWSTM0Zm1XU3JqdUYiLCJpc3MiOiJodHRwczpcL1wvZ213Z3RkY3NlbmcwMmwubmFtLm5zcm9vdC5uZXQ6OTAzMSIsImlhdCI6MTUzMTUxMTAxNCwiZXhwIjoxNTMxNTExMzE0LCJwaS5zcmkiOiJkRXVBNG55aG9WSHFxRldQUENVM1NsNFhnSUUifQ.lQ7j-PVsEJ3XvvK3KZ8V9h\_7rhvKtClutmpqoH0opVs","token\_type":"Bearer","expires\_in":1199} |

### Resource Server validates Access Token

Back channel request from Resource Server to Token Endpoint of the Authorization server for Access Token validation.

Sample Request:

|  |
| --- |
| POST /as/token.oauth2 HTTP/1.1  Content-Type: application/x-www-form-urlencoded  Host: secureaccess.sit.nam.citigroup.net  Authorization: Basic UkVTT1VSQ0VfU0VSVkVSX0NMSUVOVF9JRDpSRVNPVVJDRV9TRVJWRVJfQ0xJRU5UX1NFQ1JFVA==  Content-Length: 215  charset=UTF-8  Connection: Keep-Alive  Cache-Control: no-cache  **POSTED Content:**  grant\_type=urn:pingidentity.com:oauth2:grant\_type:validate\_bearer&token=\*\*\*\*\*\*\*\*\*  **Sample JSON server response:**  {"access\_token":{"UserName":"pp51119","OrgName":"Citigroup Inc."},"scope":"","token\_type":"urn:pingidentity.com:oauth2:validated\_token","expires\_in":294,"client\_id":"rs\_client"} |

### Client validates ID token

ID Token is a JWT Token. JWT Token is made up of the header, payload and signature. Client/UserAgent or Resource Server can validate the JWT ID Token with the following steps:

|  |  |
| --- | --- |
| Steps | Summary |
| 1 | Decrypt the token (if encrypted). CitiFed ID Tokens are not encrypted. |
| 2 | Verify the issuer claim (iss) matches the OP issuer value |
| 3 | Verify the audience claim (aud) contains the OAuth2 client\_id |
| 4 | If the token contain multiple audiences, then verify that an Authorized Party claim (azp) is present |
| 5 | If the azp claim is present, verify it matches the OAuth2 client\_id |
| 6 | Optionally verify the digital signature |
| 7 | Verify the current time is prior to the expiry claim (exp) time value |
| 8 | Client specific: Verify the token was issued within an acceptable timeframe (iat) |
| 9 | If the nonce claim (nonce) is present, verify that it matches the nonce passed in the authentication request |
| 10 | Client specific: Verify the Authn Context Reference claim (acr) value is appropriate |
| 11 | Client specific: If the authentication time claim (auth\_time) present, verify it is within an acceptable range |

### Get User Profile Attributes

Back channel request from Client to UserInfo Endpoint of the Authorization server for user profile attributes with the valid Access Token.

Sample Request:

|  |
| --- |
| GET /idp/userinfo.openid HTTP/1.1  Host: secureaccess.sit.nam.citigroup.net  Authorization: eyJhbGciOiJSUzI1NiIsImtpZCI6ImNpdGlmZWRfa2V5In0.eyJzY29wZSI6WyJvcGVuaWQiXSwiY2xpZW50X2lkIjoiYWNfb2ljX2NsaWVudCIsImlzcyI6InBmODMiLCJhdWQiOiJwcDUxMTE5Iiwic3ViIjoicHA1MTExOSIsImV4cCI6MTUzMTUyMzU5OH0.YY3fXdemRUt4HHjPtB6IYNZGrgdvMW2Ha18-GeGtkZve6ETSJIgLNC2MoObKLcsKSm\_fla-PJY-CwCHAF8t-PqtsI5OGeTcHBDMQhhQiZkloSRf6UXK-4IbeblebkCPyifJEjtGZ3leAz4JaWyf3aWnly8OA\_TaFQ\_e6KQbgADIehKAlPl48od0P7-tlBxBtLB9OAJGe-F0AtjXqx-fsqHlR4Tp7Q0h7X3p6yCEw7SNyH44u0YZCKyhg5JAyumjWovpmU7v8RjfTicmQ86X9BQtWJmGQmWtp4l7BscmyyzEylt\_hCmaQf5\_JSbHdL5W4ns8rQYOuHQnXsKaAoZNGXg  charset=UTF-8  Connection: Keep-Alive  Cache-Control: no-cache  **Sample JSON server response:**  {  "sub": "pp51119","groups":"CATE"  } |

## OpenID Connect 1.0 Implicit Client Profile Workflow

OpenID Connect Implicit Client Profile is similar to the OAuth2 Implicit grant type, but with an additional id token.



**Figure:** OpenID Connect 1.0 Basic Client Profile workflow

### Client requests and receives Access Token at Authorization Endpoint

Client/UserAgent submits the request to the Authorization Endpoint of the Authorization Server to receive AccessToken after successful user authentication followed by scope acceptance by user.

|  |  |  |
| --- | --- | --- |
| **Label** | **OAuth2 Parameter** | **Description** |
| Client ID | client\_id | Provided by SecOps |
| Client Authentication | client\_secret | Provided by SecOps |
| Response Type | response\_type | response\_type=token id\_token (and additional types in space delimited format) |
| Scope | scope | scope=openid (and additional scope values in space delimited format) |
| State | state | (optional) Value sent in this parameter is returned in the response to the Client without modification. |
| Redirect URI | redirect\_uri | Requires to be the redirect\_uri configured in Authorization Server. |
| Nonce | nonce | Value associated with client session to mitigate replay attacks. This value is passed to the ID token without modification. This value should have sufficient entropy to prevent attackers from guessing the nonce value. |

Sample Request:

|  |
| --- |
| POST /as/authorization.oauth2 HTTP/1.1  Content-Type: application/x-www-form-urlencoded  Host: secureaccess.sit.nam.citigroup.net  Content-Length: 134  Connection: Keep-Alive  Cache-Control: no-cache  **POSTED Content:**  client\_id=<CLIENT\_ID>&client\_secret=\*\*\*\*\*\*\*\*\*\*\*\*\*\*&response\_type=token%20id\_token&scope=openid&nonce=\*\*\*\*\*\*\*\*\*\*\*&state=\_\_\_\_  Server Response Sample:  {"access\_token":"eyJhbGciOiJSUzI1NiIsImtpZCI6ImNpdGlmZWRfa2V5In0.eyJzY29wZSI6WyJvcGVuaWQiXSwiY2xpZW50X2lkIjoiaW1fb2ljX2NsaWVudCIsImlzcyI6InBmODMiLCJhdWQiOiJwcDUxMTE5Iiwic3ViIjoicHA1MTExOSIsImV4cCI6MTUzMTUyNTM5OX0.OXecLNNQG4HDwD-h93blI0jeUaeRP\_tSgANXaAdYBQWIaRKCDoxuOXMAOYm0ij3IDcypZ6DXWqX\_UeJAp72P9-GBn5h8hKpuXK0eVkHWg0XAsZuXo2CPIDzhdh1xDRO\_GxHZUw-qPsyVleINF\_kuZOk2GXBgHDHIor2pWYRWGfq3keBZNEqcQl1tpvOGcaqJ7Fnzdb\_fNJGM1WxgYKivSRb-oYW5I9sJKGxLewHscewFiZe4wCA0LwBWemOitbbA\_bZKD5TlkCJOL20zFzRguoAfGyA2dsBk4EfG3YJFno7NzegnDeZ7LahTTuDFfVUK22-Jbt3MjTFZaUO7zgj7jA","id\_token":"eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJwcDUxMTE5IiwiYXVkIjoiaW1fb2ljX2NsaWVudCIsImp0aSI6InBQRVZIZXR1Z2lhVmlYOG1xMmlHdHgiLCJpc3MiOiJodHRwczpcL1wvZ213Z3RkY3NlbmcwMmwubmFtLm5zcm9vdC5uZXQ6OTAzMSIsImlhdCI6MTUzMTUyNDE5OSwiZXhwIjoxNTMxNTI0NDk5LCJwaS5zcmkiOiJkRXVBNG55aG9WSHFxRldQUENVM1NsNFhnSUUiLCJub25jZSI6IjViNWQxZDI3LWJkOTItNGQ1MS1iYzQzLWVmZmQwYzIwYjljNiIsImF0X2hhc2giOiJnRXpLTXUyRlpvUFpSZ3RfN0dUSHNBIn0.PR3PZKeT2PwL6cSioDy4HiSYkC\_TvwzQT3J\_FmyNC0k",token\_type":"Bearer","expires\_in":299} |

### Resource Server validates Access Token

Back channel request from Resource Server to Token Endpoint of the Authorization server for Access Token validation.

Sample Request:

|  |
| --- |
| POST /as/token.oauth2 HTTP/1.1  Content-Type: application/x-www-form-urlencoded  Host: secureaccess.sit.nam.citigroup.net  Authorization: Basic UkVTT1VSQ0VfU0VSVkVSX0NMSUVOVF9JRDpSRVNPVVJDRV9TRVJWRVJfQ0xJRU5UX1NFQ1JFVA==  Content-Length: 215  charset=UTF-8  Connection: Keep-Alive  Cache-Control: no-cache  **POSTED Content:**  grant\_type=urn:pingidentity.com:oauth2:grant\_type:validate\_bearer&token=\*\*\*\*\*\*\*\*\*  **Sample JSON server response:**  {"access\_token":{"UserName":"pp51119","OrgName":"Citigroup Inc."},"scope":"","token\_type":"urn:pingidentity.com:oauth2:validated\_token","expires\_in":294,"client\_id":"rs\_client"} |

### Client or Resource Server validates ID token

ID Token is a JWT Token. JWT Token is made up of the header, payload and signature. Client/UserAgent or Resource Server can validate the JWT ID Token with the following steps:

|  |  |
| --- | --- |
| Steps | Summary |
| 1 | Decrypt the token (if encrypted). CitiFed ID Tokens are not encrypted. |
| 2 | Verify the issuer claim (iss) matches the OP issuer value |
| 3 | Verify the audience claim (aud) contains the OAuth2 client\_id |
| 4 | If the token contain multiple audiences, then verify that an Authorized Party claim (azp) is present |
| 5 | If the azp claim is present, verify it matches the OAuth2 client\_id |
| 6 | Verify the digital signature |
| 7 | Verify the current time is prior to the expiry claim (exp) time value |
| 8 | Client specific: Verify the token was issued within an acceptable timeframe (iat) |
| 9 | If the nonce claim (nonce) is present, verify that it matches the nonce passed in the authentication request |
| 10 | Client specific: Verify the Authn Context Reference claim (acr) value is appropriate |
| 11 | Client specific: If the authentication time claim (auth\_time) present, verify it is within an acceptable range |
| 12 | Verify that the access token hash claim (at\_hash) matches the hash of the associated access\_token |

### Get User Profile Attributes

Back channel request from Client to UserInfo Endpoint of the Authorization server for user profile attributes with the valid Access Token (Bearer Token header).

Sample Request:

|  |
| --- |
| GET /idp/userinfo.openid HTTP/1.1  Host: secureaccess.sit.nam.citigroup.net  Authorization: eyJhbGciOiJSUzI1NiIsImtpZCI6ImNpdGlmZWRfa2V5In0.eyJzY29wZSI6WyJvcGVuaWQiXSwiY2xpZW50X2lkIjoiYWNfb2ljX2NsaWVudCIsImlzcyI6InBmODMiLCJhdWQiOiJwcDUxMTE5Iiwic3ViIjoicHA1MTExOSIsImV4cCI6MTUzMTUyMzU5OH0.YY3fXdemRUt4HHjPtB6IYNZGrgdvMW2Ha18-GeGtkZve6ETSJIgLNC2MoObKLcsKSm\_fla-PJY-CwCHAF8t-PqtsI5OGeTcHBDMQhhQiZkloSRf6UXK-4IbeblebkCPyifJEjtGZ3leAz4JaWyf3aWnly8OA\_TaFQ\_e6KQbgADIehKAlPl48od0P7-tlBxBtLB9OAJGe-F0AtjXqx-fsqHlR4Tp7Q0h7X3p6yCEw7SNyH44u0YZCKyhg5JAyumjWovpmU7v8RjfTicmQ86X9BQtWJmGQmWtp4l7BscmyyzEylt\_hCmaQf5\_JSbHdL5W4ns8rQYOuHQnXsKaAoZNGXg  charset=UTF-8  Connection: Keep-Alive  Cache-Control: no-cache  **Sample JSON server response:**  {  "sub": "pp51119","groups":"CATE"  } |

## OAuth2 Token Refresh

A Refresh Token can be used to request new Access Token (and Refresh Token) and avoid user re-authentication. If the Refresh Token is invalidated due to one of the following reasons, the user should re-authenticate to according to the grant type to get new Access Token and Refresh Token.

1. Refresh Token expiration
2. User revoked the Refresh Token
3. Refresh Token administratively revoked

Requesting a token refresh with additional scopes will result in an error. Omitting the scope will result in the token refresh at the same scope. Sub-scope token refresh results in Access Token for the sub-scope.



**Figure**: Token Refresh workflow

|  |  |  |
| --- | --- | --- |
| **Label** | **OAuth2 Parameter** | **Description** |
| Client ID | client\_id | Provided by SecOps |
| Client Authentication | client\_secret | Provided by SecOps |
| Allowed Grant Type | response\_type  grant\_type | grant\_type=refresh\_token |
| Scope | scope | Scope of the request from client. This pre-defined scope is configured in the Authorization Server. User accepts the scope to give permission to the Client/UserAgent after successful authentication. |
| Refresh Token | refresh\_token | Value of the Refresh Token |

Sample Request:

|  |
| --- |
| POST /as/token.oauth2 HTTP/1.1  Content-Type: application/x-www-form-urlencoded  Host: secureaccess.sit.nam.citigroup.net  Authorization: Basic PFJPX0NMSUVOVF9JRD46PFJPX0NMSUVOVF9QQVNTV09SRD4=  Content-Length: 134  Connection: Keep-Alive  Cache-Control: no-cache  **POSTED Content:**  grant\_type=refresh\_token&refresh\_token=\*\*\*\*\*\*\*  Server Response Sample:  {"access\_token":"5FjxuAlVi6XDTkCckxvouxpSOrUi","refresh\_token":"Q5hauS312YeL5WrIcT8jHhJ5fgyCukb3GiCln7ajqE","token\_type":"Bearer","expires\_in":299} |

## Grant Management – Revocation of Refresh Tokens

Refresh Tokens and Internally managed Access tokens can be revoked by submitting the request to the revocation endpoint of the CitiFed OAuth2 Authorization Server. JWT Access Tokens can be indirectly revoked by revoking the Refresh Token.

The user can revoke the refresh tokens by accessing the



***Figure***: Refresh Token Revocation workflow

### Client revokes Refresh Token at Token Revocation Endpoint

Client/UserAgent submits the request to the Token Revocation Endpoint for revoking the Refresh Token.

|  |  |  |
| --- | --- | --- |
| **Label** | **OAuth2 Parameter** | **Description** |
| Client ID | client\_id | Provided by SecOps |
| Client Authentication | client\_secret | Provided by SecOps |
| Token Type Hint | token\_type\_hint | token\_type\_hint=refresh\_token should be used for revoking refresh tokens. For internally managed access tokens, token\_type\_hint=access\_token can be used. |
| Token | token | refresh token value |

Sample Request:

|  |
| --- |
| POST /as/revoke\_token.oauth2 HTTP/1.1  Content-Type: application/x-www-form-urlencoded  Host: secureaccess.sit.nam.citigroup.net  Authorization: Basic PFJPX0NMSUVOVF9JRD46PFJPX0NMSUVOVF9QQVNTV09SRD4=  Content-Length: 134  Connection: Keep-Alive  Cache-Control: no-cache  **POSTED Content:**  token\_type\_hint=refresh\_token&token=STM64qQSEP6SlOki4W0lWR3KNosR7idRytyFOuXyso  Server Response Sample:  HTTP 200 OK  Headers:  **Cache-Control →**no-cache, no-store  **Content-Security-Policy →**referrer origin  **Content-Type →**text/html;charset=utf-8  **Date →**Wed, 11 Jul 2018 19:27:42 GMT  **Expires →**Thu, 01 Jan 1970 00:00:00 GMT  **Pragma →**no-cache  **Transfer-Encoding →**chunked  **X-Frame-Options →**SAMEORIGIN |

## Grant Management by user

Refresh Tokens can be revoked by the user by accessing the Grant management URL of CitiFed after user authentication with SSO credentials.

|  |  |
| --- | --- |
| **Label** | **Description** |
| Grant Management URL | <https://secureaccess.nam.citigroup.net/as/oauth_access_grants.ping> |
| User Authentication | SSO user credentials |



Figure: Sample page with all Refresh Tokens of the user.

## Client/UserAgent client\_secret update with OTP from Ops

TBD

# Client Integration Requirements

## WS-Trust integration

CitiFederation provides web services clients with WS-Trust for processing SOAP requests. CitiFederation WS-Trust requires 2-way SSL (client authenticated SSL) for processing requests.

IDP WS-Trust clients include a valid SMSESSION cookie value or OpenToken as binary token in the RST Soap request.

SP WS-Trust clients include a valid SAML 2.0 Assertion and the Assertion ID in the RST Soap request.

Java and .NET clients can encapsulate the Web Service client code with SOAP parsers and other API for processing ease. A sample java HTTP Client code is provided in the Appendix **Section 8**.

### TrustStore & Client Authenticated SSL Setup

Import the CitiFederation Web Server Cert’s signer CA certificates into an SSL Trust keystore. This trust store will be used for verifying the server certificate for 2-way SSL.

Send the client certificate’s signer CA cert & the client certificate DN to CitiFederation for configuration of 2-way SSL.

### Java based Clients

Clients using JAVA can either use HTTP Client code or JAX-RPC/JAX-WS based clients to connect to the CitiFederation web services. A sample HTTPS Client code is provided in Appendix **Section 8**.

## OpenToken integration

OpenToken creation or consumption requires the following jar files (or higher versions) to present in the classpath.

|  |
| --- |
| opentoken-agent-2.5.1.jar  commons-collections-3.2.jar  commons-beanutils.jar  commons-logging.jar |

These jar file location for download is available in **Section 1.3**.

**User Attributes in OpenToken**

|  |  |  |  |
| --- | --- | --- | --- |
| **Attribute Name** | **Attribute Value** | **Description** | **Mandatory** |
| SUBJECT | UserID | A valid UserID should be used for SUBJECT attribute’s value. | YES |
| <NAME> | <VALUE> | Other user attributes | NO |

There are two types of OpenToken processing use cases.

1. Create OpenToken with name/value pairs

2. Read OpenToken

The sample java code for creating or reading OpenTokens is available in **Section 7**.

# APPENDIX: SOAP Request-Response Samples

This section provides SOAP Request samples for the CitiFederation WS-Trust.

## IDP WS-Trust Samples

**IDP RST SOAP Request Sample:**

The following variables should be replaced in the RSTs.

|  |  |
| --- | --- |
| **String to replace** | **Value Description** |
| REPLACE\_ID\_<NUMBER> | These strings should be in the format mentioned in the [XML datatype ID](http://www.w3.org/TR/2004/REC-xmlschema-2-20041028/datatypes.html#ID)  CitiFederation Engineering recommends using a 28 random character value for REPLACE\_ID\_<NUMBER> values. |
| CREATE\_TIMESTAMP | This string should have the format YYYY-MM-DD”T”HH:MM:SS.<ms>”Z” (UTC time). Sample: 2014-06-23T16:22:39.834Z |
| EXPIRES\_TIMESTAMP | should have the format YYYY-mm-DD”T”HH:MM:SS.SSS”Z” (UTC time). Sample: 2014-06-23T16:22:44.834Z |
| BINARY\_TOKEN\_TYPE | WAM Token type (SMSESSION) is urn:pingidentity:wam  OpenToken Token Type is urn:pingidentity:opentoken |
| TOKEN\_VALUE | This string should have the token value in ASCII format. |

NOTE: The string literal “#” is used with the REPLACE\_ID\_3 in the IDP RST sample below, hence do not substitute the string literal “#”. Sample in RST: URI="#REPLACE\_ID\_3"

|  |
| --- |
| <soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">  <soap:Header>  <wsa:Action xmlns:wsa="http://www.w3.org/2005/08/addressing" xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd" wsu:Id="REPLACE\_ID\_1">http://docs.oasis-open.org/ws-sx/ws-trust/200512/RST/Issue</wsa:Action>  <wsse:Security xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd" soap:mustUnderstand="1">  <wsu:Timestamp xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd" wsu:Id="REPLACE\_ID\_2">  <wsu:Created>CREATE\_TIMESTAMP</wsu:Created>  <wsu:Expires>EXPIRES\_TIMESTAMP</wsu:Expires>  </wsu:Timestamp>  <wsse:BinarySecurityToken xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd" EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0#Base64Binary" ValueType="BINARY\_TOKEN\_TYPE" wsu:Id="REPLACE\_ID\_3">TOKEN\_VALUE</wsse:BinarySecurityToken>  </wsse:Security>  </soap:Header>  <soap:Body xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd" wsu:Id="REPLACE\_ID\_4">  <wst:RequestSecurityToken xmlns:wst="http://docs.oasis-open.org/ws-sx/ws-trust/200512/">  <wst:RequestType>http://docs.oasis-open.org/ws-sx/ws-trust/200512/Issue</wst:RequestType>  <wsp:AppliesTo xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy">  <wsa:EndpointReference xmlns:wsa="http://www.w3.org/2005/08/addressing">  <wsa:Address>rptest</wsa:Address>  </wsa:EndpointReference>  </wsp:AppliesTo>  <wst:OnBehalfOf>  <wsse:SecurityTokenReference xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">  <wsse:Reference URI="#REPLACE\_ID\_3" ValueType="urn:pingidentity:wam"/>  </wsse:SecurityTokenReference>  </wst:OnBehalfOf>  </wst:RequestSecurityToken>  </soap:Body>  </soap:Envelope> |

**IDP RSTR SOAP Response Sample:**

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?>  <S11:Envelope xmlns:S11="http://schemas.xmlsoap.org/soap/envelope/">  <S11:Header>  <add:To xmlns:add="http://www.w3.org/2005/08/addressing">http://schemas.xmlsoap.org/ws/2004/08/addressing/role/anonymous</add:To>  <add:Action xmlns:add="http://www.w3.org/2005/08/addressing">http://docs.oasis-open.org/ws-sx/ws-trust/200512/RSTRC/IssueFinal</add:Action>  <wsse:Security xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd" S11:mustUnderstand="1">  <wsu:Timestamp xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd" wsu:Id="f87f2019-8bf1-495f-9544-7c0e51cdcc72">  <wsu:Created>2014-06-20T15:10:03.832Z</wsu:Created>  <wsu:Expires>2014-06-20T15:15:03.832Z</wsu:Expires>  </wsu:Timestamp>  </wsse:Security>  </S11:Header>  <S11:Body>  <ns:RequestSecurityTokenResponseCollection xmlns:ns="http://docs.oasis-open.org/ws-sx/ws-trust/200512/">  <ns:RequestSecurityTokenResponse>  <ns:TokenType>http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-1.1#SAMLV2.0</ns:TokenType>  <ns:RequestedSecurityToken>  <saml:Assertion xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion" ID="mtqFMeoHcRggF9lElylTEp9l\_aS" IssueInstant="2014-06-20T15:10:03.810Z" Version="2.0">  <saml:Issuer>demows-idp</saml:Issuer>  <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">  <ds:SignedInfo>  <ds:CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>  <ds:SignatureMethod Algorithm="http://www.w3.org/2001/04/xmldsig-more#rsa-sha512"/>  <ds:Reference URI="#mtqFMeoHcRggF9lElylTEp9l\_aS">  <ds:Transforms>  <ds:Transform Algorithm="http://www.w3.org/2000/09/xmldsig#enveloped-signature"/>  <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>  </ds:Transforms>  <ds:DigestMethod Algorithm="http://www.w3.org/2001/04/xmlenc#sha512"/>  <ds:DigestValue>OaqJ++GNBMOTU3CWAlMT+RboLoEXCeJgzbUDgE778a43nEE1dWKMvYx98fcyLd83zW3tX2Ss3pSW  YKKWZN++rA==</ds:DigestValue>  </ds:Reference>  </ds:SignedInfo>  <ds:SignatureValue>  jXqU/SVNbbT5Ba1Ge0bN9I0jqXEFSPdk522OWLOvSyvcTRUYRMlapUpWoeaWcEP9ObhO8jqKI/DA  jjaulLmg2wrBZeguPxXcgmp79wQcTg3CXvcbZLUrVJluu67oxPWoJUjxmL3OV0Dn4xSPpWf2Jw7l  TpDgPj4WCaHU8gsi7e9715OJH3kRWIYSXbH9+tUIGWFc/mRTlyGTE9YYJmShZDaTU/RkAjpKuV7q  HYVssl764YNQZzhNxQZQ1vbN/qJ9YFbgXAA5YfTuHwDPLtscSRJz5fMgVZrHREmnaBTB+i6+vb7m  jP6afanZ1hPDJRHWISjc7LvZbf0GNIW9rjq1xA==  </ds:SignatureValue>  <ds:KeyInfo>  <ds:X509Data>  <ds:X509Certificate>  MIIDxDCCAqygAwIBAgIJAJvlGt6alpadMA0GCSqGSIb3DQEBBQUAMFsxCzAJBgNVBAYTAlVTMRAw  DgYDVQQIDAdGbG9yaWRhMRIwEAYDVQQKDAlDaXRpZ3JvdXAxDTALBgNVBAsMBENBVEUxFzAVBgNV  BAMMDkNpdGlGZWRlcmF0aW9uMB4XDTE0MDIxMzE1MTU0OVoXDTE2MDgwMTE1MTU0OVowazELMAkG  A1UEBhMCVVMxGDAWBgNVBAoMD0NpdGlncm91cCBRQUNBMTELMAkGA1UECwwCTkExDTALBgNVBAsM  BENBTEMxJjAkBgNVBAMMHWdtd213ZGNzZW5nMDJsLm5hbS5uc3Jvb3QubmV0MIIBIjANBgkqhkiG  9w0BAQEFAAOCAQ8AMIIBCgKCAQEAvPRp+aJS7gEypuPzSpVLFOxSXER1GpYQeztFJuG1ugWTH0Xz  ehl+PYuauKMNsypZritXF5rZyTAqjPBBRFQU+5fJt9DB8whHpHPaJ/olcD59hYexY3J4qEiliAOS  PC7at6AfEPe9zD4bL3HtUrqYVa+hg+AtO/xAkmbPeVGVUBJQdgOVEJZSDWEHRthT+Z5CGnFo7Tqk  jWijUHUGALteilRR3lCe69b2w3lk5qQ/9puI11XFsqzqjXuIPBoAQUZThwr2ne/AWka6UvU+JgMJ  ngFaReMdiN7bDlRDK3dv3wL3/wXmBcWY/kk+f0pEfcxkmAgTN+vyawmUjAUERKRxxQIDAQABo3sw  eTAJBgNVHRMEAjAAMCwGCWCGSAGG+EIBDQQfFh1PcGVuU1NMIEdlbmVyYXRlZCBDZXJ0aWZpY2F0  ZTAdBgNVHQ4EFgQULgxlFXD2bNu+vLA42U9JJ44tQF0wHwYDVR0jBBgwFoAUDH6hJ2jxaDBeJSjW  z7a8zy8sQwwwDQYJKoZIhvcNAQEFBQADggEBALyq9zCkGAHTMUpTxeOMStJHC36dPMMrXdRgikNb  sE6zibL5o+yqFz4LfuLELa8tHeVZ8Ks3NLkNEo+Xc+tgmAdHB57Uj4PQMhR3jTP8j5wXaO9okHdG  2r6sf2CB6BOwyMp9c/IurwiLp1MaR7KRGgRIvpZ8TxqNQvrdwYK+xa2qPL6rChKS35zlJBhwVbce  P30mjGuGE0opp6I2H0IN3uEWYNDWe8tb9+rc6yzuwrPeFY0wvdrVvhJ2t6CjmNzBvzVn8qzwPLcS  7NlDPXEcHC5JCoNMmW2+X3hkIqZkEjwcGOA+u8HfWtjg0vRs5QPdStF6Qox0ofeSMIKG2bFseD0=  </ds:X509Certificate>  </ds:X509Data>  <ds:KeyValue>  <ds:RSAKeyValue>  <ds:Modulus>  vPRp+aJS7gEypuPzSpVLFOxSXER1GpYQeztFJuG1ugWTH0Xzehl+PYuauKMNsypZritXF5rZyTAq  jPBBRFQU+5fJt9DB8whHpHPaJ/olcD59hYexY3J4qEiliAOSPC7at6AfEPe9zD4bL3HtUrqYVa+h  g+AtO/xAkmbPeVGVUBJQdgOVEJZSDWEHRthT+Z5CGnFo7TqkjWijUHUGALteilRR3lCe69b2w3lk  5qQ/9puI11XFsqzqjXuIPBoAQUZThwr2ne/AWka6UvU+JgMJngFaReMdiN7bDlRDK3dv3wL3/wXm  BcWY/kk+f0pEfcxkmAgTN+vyawmUjAUERKRxxQ==  </ds:Modulus>  <ds:Exponent>AQAB</ds:Exponent>  </ds:RSAKeyValue>  </ds:KeyValue>  </ds:KeyInfo>  </ds:Signature>  <saml:Subject>  <saml:NameID Format="urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified">pp51119</saml:NameID>  <saml:SubjectConfirmation Method="urn:oasis:names:tc:SAML:2.0:cm:bearer"/>  </saml:Subject>  <saml:Conditions NotBefore="2014-06-20T15:05:03.811Z" NotOnOrAfter="2014-06-20T15:40:03.811Z">  <saml:AudienceRestriction>  <saml:Audience>demows-sp</saml:Audience>  </saml:AudienceRestriction>  </saml:Conditions>  <saml:AuthnStatement AuthnInstant="2014-06-20T15:10:03.810Z" SessionIndex="mtqFMeoHcRggF9lElylTEp9l\_aS">  <saml:AuthnContext>  <saml:AuthnContextClassRef>urn:oasis:names:tc:SAML:2.0:ac:classes:unspecified</saml:AuthnContextClassRef>  </saml:AuthnContext>  </saml:AuthnStatement>  </saml:Assertion>  </ns:RequestedSecurityToken>  <ns:Lifetime>  <wsu:Created xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd">2014-06-20T15:10:03.810Z</wsu:Created>  <wsu:Expires xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd">2014-06-20T15:40:03.811Z</wsu:Expires>  </ns:Lifetime>  <ns:RequestedAttachedReference>  <wsse:SecurityTokenReference xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd" xmlns:wsse11="http://docs.oasis-open.org/wss/oasis-wss-wssecurity-secext-1.1.xsd" wsse11:TokenType="http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-1.1#SAMLV2.0">  <wsse:KeyIdentifier ValueType="http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-1.1#SAMLID">mtqFMeoHcRggF9lElylTEp9l\_aS</wsse:KeyIdentifier>  </wsse:SecurityTokenReference>  </ns:RequestedAttachedReference>  <wsp:AppliesTo xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy">  <add:EndpointReference xmlns:add="http://www.w3.org/2005/08/addressing">  <add:Address>rptest</add:Address>  </add:EndpointReference>  </wsp:AppliesTo>  </ns:RequestSecurityTokenResponse>  </ns:RequestSecurityTokenResponseCollection>  </S11:Body>  </S11:Envelope> |

## User Authentication

**UserAuthentication RST SOAP Request Sample:**

The following variables should be replaced in the RSTs.

|  |  |
| --- | --- |
| **String to replace** | **Value Description** |
| REPLACE\_ID\_<NUMBER> | These strings should be in the format mentioned in the [XML datatype ID](http://www.w3.org/TR/2004/REC-xmlschema-2-20041028/datatypes.html#ID)  CitiFederation Engineering recommends using a 28 random character value for REPLACE\_ID\_<NUMBER> values. |
| CREATE\_TIMESTAMP | This string should have the format YYYY-MM-DD”T”HH:MM:SS.<ms>”Z” (UTC time). Sample: 2014-06-23T16:22:39.834Z |
| EXPIRES\_TIMESTAMP | should have the format YYYY-mm-DD”T”HH:MM:SS.SSS”Z” (UTC time). Sample: 2014-06-23T16:22:44.834Z |
| BINARY\_TOKEN\_TYPE | WAM Token type (SMSESSION) is urn:pingidentity:wam |
| TOKEN\_VALUE | This value is Base64Encode(USERID+”|”+PASSWORD) where USERID and PASSWORD are the CitiSSO user credentials. |

**UserAuthentication RST SOAP Request Sample:**

|  |
| --- |
| <soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">  <soap:Header>  <wsa:Action xmlns:wsa="http://www.w3.org/2005/08/addressing" xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd" wsu:Id="REPLACE\_ID\_1">http://docs.oasis-open.org/ws-sx/ws-trust/200512/RST/Issue</wsa:Action>  <wsse:Security xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd" soap:mustUnderstand="1">  <wsu:Timestamp xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd" wsu:Id="REPLACE\_ID\_2">  <wsu:Created>CREATE\_TIMESTAMP</wsu:Created>  <wsu:Expires>EXPIRES\_TIMESTAMP</wsu:Expires>  </wsu:Timestamp>  <wsse:BinarySecurityToken xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd" EncodingType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0#Base64Binary" ValueType="BINARY\_TOKEN\_TYPE" wsu:Id="REPLACE\_ID\_3">TOKEN\_VALUE</wsse:BinarySecurityToken>  </wsse:Security>  </soap:Header>  <soap:Body xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd" wsu:Id="REPLACE\_ID\_4">  <wst:RequestSecurityToken xmlns:wst="http://docs.oasis-open.org/ws-sx/ws-trust/200512/">  <wst:RequestType>http://docs.oasis-open.org/ws-sx/ws-trust/200512/Issue</wst:RequestType>  <wsp:AppliesTo xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy">  <wsa:EndpointReference xmlns:wsa="http://www.w3.org/2005/08/addressing">  <wsa:Address>rptest</wsa:Address>  </wsa:EndpointReference>  </wsp:AppliesTo>  <wst:OnBehalfOf>  <wsse:SecurityTokenReference xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd">  <wsse:Reference URI="#REPLACE\_ID\_3" ValueType="urn:pingidentity:wam"/>  </wsse:SecurityTokenReference>  </wst:OnBehalfOf>  </wst:RequestSecurityToken>  </soap:Body>  </soap:Envelope> |

**UserAuthentication RSTR SOAP Response Sample:**

|  |
| --- |
| <S11:Envelope xmlns:S11="http://schemas.xmlsoap.org/soap/envelope/"><S11:Header><add:To xmlns:add="http://www.w3.org/2005/08/addressing">http://schemas.xmlsoap.org/ws/2004/08/addressing/role/anonymous</add:To><add:Action xmlns:add="http://www.w3.org/2005/08/addressing">http://docs.oasis-open.org/ws-sx/ws-trust/200512/RSTRC/IssueFinal</add:Action><wsse:Security S11:mustUnderstand="1" xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd"><wsu:Timestamp wsu:Id="e8478f8f-e00d-4127-a805-56add4fe822f" xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"><wsu:Created>2017-04-21T04:08:08.839Z</wsu:Created><wsu:Expires>2017-04-21T04:13:08.839Z</wsu:Expires></wsu:Timestamp></wsse:Security></S11:Header><S11:Body><ns:RequestSecurityTokenResponseCollection xmlns:ns="http://docs.oasis-open.org/ws-sx/ws-trust/200512/"><ns:RequestSecurityTokenResponse><ns:TokenType>urn:pingidentity:wam</ns:TokenType><ns:RequestedSecurityToken><wsse:BinarySecurityToken EncodingType="openEncodingType" wsu:Id="JjPL\_Ywes8jdUxNqjWQV\_d0a6LC" ValueType="urn:pingidentity:wam" xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd" xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"></wsse:BinarySecurityToken></ns:RequestedSecurityToken></ns:RequestSecurityTokenResponse></ns:RequestSecurityTokenResponseCollection></S11:Body></S11:Envelope> |

NOTE: The highlighted text above is the SMSESSION value.

## ChangePassword Sample:

HTTP POST parameters to be submitted for ChangePassword RESTful call:

|  |  |
| --- | --- |
| **HTTP POST parameters** | **Value Description** |
| username | UserID value |
| oldPassword | URL encoded value of user’s old password |
| newPassword | URL encoded value of user’s new password |

Custom Request Header for ChangePassword RESTful call:

|  |  |
| --- | --- |
| **Request Header Name** | **Request Header Value** |
| CitiFedCPHeader | yes |

Response of ChangePassword RESTful call:

|  |  |  |
| --- | --- | --- |
| **Response contains** | **Use case** | **Comments** |
| SMSESSION value | Positive | The ChangePassword is successful if the response contains a valid SMSESSION value |
| INVALID | Negative | The ChangePassword failed for the user if the response contains the string literal “INVALID”. Check with Operations team for error details. |

## SP WS-Trust Samples

**SP RST SOAP Request Sample:**

The following variables should be replaced in the RSTs.

|  |  |
| --- | --- |
| **String to replace** | **Value Description** |
| REPLACE\_ID\_<NUMBER> | These strings should be in the format mentioned in the [XML datatype ID](http://www.w3.org/TR/2004/REC-xmlschema-2-20041028/datatypes.html#ID)  CitiFederation Engineering recommends using a 28 random character value for REPLACE\_ID\_<NUMBER> values. |
| CREATE\_TIMESTAMP | This string should have the format YYYY-MM-DD”T”HH:MM:SS.<ms>”Z” (UTC time). Sample: 2014-06-23T16:22:39.834Z |
| EXPIRES\_TIMESTAMP | should have the format YYYY-mm-DD”T”HH:MM:SS.SSS”Z” (UTC time). Sample: 2014-06-23T16:22:44.834Z |
| SAML\_2.0\_ASSERTION | This string should be replaced with a valid SAML 2.0 Assertion. |
| SAML\_2.0\_ASSERTION\_ID | This string should be replaced with the SAML 2.0 Assertion’s ID parsed from the SAML 2.0 Assertion. |

|  |
| --- |
| <soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">  <soap:Header>  <wsa:Action xmlns:wsa="http://www.w3.org/2005/08/addressing" xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd" wsu:Id="REPLACE\_ID\_1">http://docs.oasis-open.org/ws-sx/ws-trust/200512/RST/Issue</wsa:Action>  <wsse:Security xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd" soap:mustUnderstand="1">  <wsu:Timestamp xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd" wsu:Id="REPLACE\_ID\_2">  <wsu:Created>CREATE\_TIMESTAMP</wsu:Created>  <wsu:Expires>EXPIRES\_TIMESTAMP</wsu:Expires>  </wsu:Timestamp>  SAML\_2.0\_ASSERTION  </wsse:Security>  </soap:Header>  <soap:Body xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd" wsu:Id="REPLACE\_ID\_3">  <wst:RequestSecurityToken xmlns:wst="http://docs.oasis-open.org/ws-sx/ws-trust/200512/">  <wst:TokenType>urn:pingidentity:wam</wst:TokenType>  <wst:RequestType>http://docs.oasis-open.org/ws-sx/ws-trust/200512/Issue</wst:RequestType>  <wst:ValidateTarget>  <wsse:SecurityTokenReference xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd" xmlns:wsse11="http://docs.oasis-open.org/wss/oasis-wss-wssecurity-secext-1.1.xsd" wsse11:TokenType="http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-1.1#SAMLV2.0">  <wsse:KeyIdentifier ValueType="http://docs.oasis-open.org/wss/oasis-wss-saml-token-profile-1.1#SAMLID">SAML\_2.0\_ASSERTION\_ID</wsse:KeyIdentifier>  </wsse:SecurityTokenReference>  </wst:ValidateTarget>  </wst:RequestSecurityToken>  </soap:Body>  </soap:Envelope> |

**SP RSTR Response with WAM (SMSESSION) Sample:**

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?>  <S11:Envelope xmlns:S11="http://schemas.xmlsoap.org/soap/envelope/">  <S11:Header>  <add:To xmlns:add="http://www.w3.org/2005/08/addressing">http://schemas.xmlsoap.org/ws/2004/08/addressing/role/anonymous</add:To>  <add:Action xmlns:add="http://www.w3.org/2005/08/addressing">http://docs.oasis-open.org/ws-sx/ws-trust/200512/RSTRC/IssueFinal</add:Action>  <wsse:Security xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd" S11:mustUnderstand="1">  <wsu:Timestamp xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd" wsu:Id="87a490c4-bbf8-44bd-90e4-52e0788f89d3">  <wsu:Created>2014-06-20T15:28:32.852Z</wsu:Created>  <wsu:Expires>2014-06-20T15:33:32.852Z</wsu:Expires>  </wsu:Timestamp>  </wsse:Security>  </S11:Header>  <S11:Body>  <ns:RequestSecurityTokenResponseCollection xmlns:ns="http://docs.oasis-open.org/ws-sx/ws-trust/200512/">  <ns:RequestSecurityTokenResponse>  <ns:TokenType>urn:pingidentity:wam</ns:TokenType>  <ns:RequestedSecurityToken>  <wsse:BinarySecurityToken xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-secext-1.0.xsd" xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd" EncodingType="openEncodingType" wsu:Id="eNbK\_M.jZ-7yImuQvEo-rKFt3q6" ValueType="urn:pingidentity:wam"></wsse:BinarySecurityToken>  </ns:RequestedSecurityToken>  </ns:RequestSecurityTokenResponse>  </ns:RequestSecurityTokenResponseCollection>  </S11:Body>  </S11:Envelope> |

# APPENDIX: Java sample code for OpenToken

The following is a sample java code for reading or creating OpenToken.

|  |  |  |
| --- | --- | --- |
| **Variable Name** | **Value** | **Comments** |
| CIPHER\_SUITE | Token.CIPHER\_SUITE\_AES128CBC  or  Token.CIPHER\_SUITE\_AES256CBC  or  Token.CIPHER\_SUITE\_NULL |  |
| COOKIE\_DOMAIN | .citigroup.net or .citigroup.com |  |
| COOKIE\_PATH | / |  |
| TOKEN\_NOTBEFORE\_TOLERANCE | 60 |  |
| OBFUSCATE\_PASSWORD | false |  |
| TOKEN\_RENEW\_UNTIL | 43200 |  |
| SECURE\_COOKIE | false |  |
| SESSION\_COOKIE | false |  |
| TOKEN\_LIFETIME | 300 |  |
| USE\_COOKIE | false |  |
| USE\_SUNJCE | false |  |
| OPEN\_TOKEN\_NAME | IDP use case: citifedidpot  SP use case: citifedspot | Two separate values for IDP or SP use cases. |
| OPEN\_TOKEN\_PASSWORD | Password1 | Literal default value used by CitiFederation |

**Steps**

1. Import packages

|  |
| --- |
| import com.pingidentity.opentoken.Agent;  import com.pingidentity.opentoken.Token;  import com.pingidentity.opentoken.AgentConfiguration;  import com.pingidentity.opentoken.TokenException;  import com.pingidentity.opentoken.util.UrlHelper;  import org.apache.commons.collections.MultiMap;  import org.apache.commons.collections.map.MultiValueMap; |

1. Initialize Agent in the class

|  |
| --- |
| public static Agent agentToCreateOpenToken = null;  public static Agent agentToReadOpenToken = null;  public static AgentConfiguration configureAgent() throws Exception {  AgentConfiguration config = new AgentConfiguration();  config.setCipherSuite( CIPHER\_SUITE);  config.setCookieDomain(COOKIE\_DOMAIN);  config.setCookiePath(COOKIE\_PATH);  config.setNotBeforeTolerance(TOKEN\_NOTBEFORE\_TOLERANCE);  config.setObfuscatePasword(OBFUSCATE\_PASSWORD);  //SETTING DEFAULT PASSWORD FOR OpenToken created by CitiFederation instances  config.setPassword(OPEN\_TOKEN\_PASSWORD);  config.setRenewUntilLifetime(TOKEN\_RENEW\_UNTIL);  config.setSecureCookie(SECURE\_COOKIE);  config.setSessionCookie(SESSION\_COOKIE);  config.setTokenLifetime(TOKEN\_LIFETIME);  config.setTokenName(OPEN\_TOKEN\_NAME);  config.setUseCookie(USE\_COOKIE);  config.setUseSunJCE(USE\_SUNJCE);  return config;  } |

1. Main code segment for creating OpenToken

|  |
| --- |
| ////CREATE OpenToken  if (agentToCreateOpenToken == null) {  agentToCreateOpenToken = new Agent(configureAgent());  }  //SOEID sample used as username  String username = “pp51119”;  Map<String, String> userInfo = new HashMap<String, String>();  userInfo.put(Agent.TOKEN\_SUBJECT, username);  //Additional attributes  userInfo.put("name",name);  userInfo.put("email", mail);  //MultValue attribute  userInfo.put(“group”,“engineeringGroup”);  userInfo.put(“group”, “tampaGroup”);  //Create OpenToken using Agent  openToken = agentToCreateOpenToken.writeToken(userInfo); |

1. Main code segment for reading OpenToken

|  |
| --- |
| ////READ OpenToken  if (agentToReadOpenToken == null) {  agentToReadOpenToken = new Agent(configureAgent());  }  //Get OpenToken to read  String openToken = readOpenToken();  try {  //PARSE OPEN TOKEN TO GET MAP  Map<Object, String> userData = agentToReadOpenToken.readToken(openToken);  Iterator iter = (Iterator) userData.keySet().iterator();  while(iter.hasNext()) {  String paramName = (String) iter.next();  Object paramValue = userData.get(paramName);  if (paramValue instanceof String) {  //SINGLE VALUE ATTRIBUTES  String paramValueString = (String) paramValue;  //System.out.println("FROM OPEN TOKEN: NAME["+paramName +"] = "+ paramValueString);  } else if (paramValue instanceof List) {  //MULTI VALUE ATTRIBUTES  List paramValueList = (List) paramValue;  Iterator listIter = (Iterator) paramValueList.iterator();  while (listIter.hasNext()) {  String listParamValue = (String) listIter.next();  //System.out.println("FROM OPEN TOKEN: LIST\_NAME["+paramName +"] = "+ listParamValue);  }  }  }  } catch (TokenException te) {  System.out.println(“Token Exception while parsing open token value with exception: \n”+te.getMessage());  te.printStackTrace();  } |

# APPENDIX: Java sample code for WS-Trust

This section contains a sample HTTP Client code for submitting WS-Trust RST requests to CitiFederation instance.

The following variables should be replaced in the code.

|  |  |  |
| --- | --- | --- |
| **Variable Name** | **Value** | **Comments** |
| KEYSTORE\_LOCATION | <full path to keystore file> |  |
| KEYSTORE\_TYPE | PKCS12 or JKS | One of the two values |
| KEYSTORE\_PASSWORD | <password character array> |  |
| TRUSTSTORE\_LOCATION | <full path to truststore file> |  |
| TRUSTSTORE\_TYPE | “PKCS12” or “JKS” | One of the two values |
| TRUSTSTORE\_PASSWORD | <password character array> |  |
| WSTRUST\_IDP\_URL |  | URL to be provided per env. |
| WSTRUST\_SP\_URL |  | URL to be provided per env. |
| RST\_REQUEST |  |  |
|  |  |  |
| TOKEN\_VALUE | SMSESSION value or OpenToken value or SAML 2.0 Assertion | One of the three values |
| CONNECTION\_READ\_TIMEOUT |  | Integer value in seconds |
|  |  |  |
|  |  |  |

1. Import packages for generic HTTP Client

|  |
| --- |
| import javax.net.ssl.HttpsURLConnection;  import javax.net.ssl.KeyManager;  import javax.net.ssl.KeyManagerFactory;  import javax.net.ssl.SSLContext;  import javax.net.ssl.SSLSocketFactory;  import javax.net.ssl.TrustManager;  import javax.net.ssl.TrustManagerFactory;  import javax.net.ssl.X509KeyManager;  import javax.xml.soap.SOAPConnection;  import javax.xml.soap.SOAPConnectionFactory;  import java.io.\*;  import java.net.Socket;  import java.net.SocketTimeoutException;  import java.net.URL;  import java.util.\*;  import java.security.KeyManagementException;  import java.security.KeyStore;  import java.security.KeyStoreException;  import java.security.NoSuchAlgorithmException;  import java.security.Principal;  import java.security.PrivateKey;  import java.security.UnrecoverableKeyException;  import java.security.cert.CertificateException;  import java.security.cert.X509Certificate; |

1. Import Keystore and Truststore for 2-way SSL

|  |
| --- |
| SSLSocketFactory factory = null;  try {  //create key and trust managers  //create Inputstream to keystore file  java.io.InputStream inputStream = new java.io.FileInputStream(KEYSTORE\_LOCATION);  //create keystore object, load it with keystorefile data  KeyStore keyStore = KeyStore.getInstance(KEYSTORE\_TYPE);  keyStore.load(inputStream, KEYSTORE\_PASSWORD == null ? null : KEYSTORE\_PASSWORD);  KeyManager[] keyManagers = null;  //create keymanager factory and load the keystore object in it  KeyManagerFactory keyManagerFactory = KeyManagerFactory.getInstance(KeyManagerFactory.getDefaultAlgorithm());  keyManagerFactory.init(keyStore, KEYSTORE\_PASSWORD == null ? null : KEYSTORE\_PASSWORD);  managers = keyManagerFactory.getKeyManagers();  //create Inputstream to truststore file  java.io.InputStream inputStream = new java.io.FileInputStream(TRUSTSTORE\_LOCATION);  //create keystore object, load it with truststorefile data  KeyStore trustStore = KeyStore.getInstance(TRUSTSTORE\_TYPE);  trustStore.load(inputStream, TRUSTSTORE\_PASSWORD == null ? null : TRUSTSTORE\_PASSWORD);  //create trustmanager factory and load the keystore object in it  TrustManagerFactory trustManagerFactory = TrustManagerFactory.getInstance(TrustManagerFactory.getDefaultAlgorithm());  trustManagerFactory.init(trustStore);  TrustManager[] trustManagers = trustManagerFactory.getTrustManagers();  //init context with managers data  SSLContext context = SSLContext.getInstance("SSLv3");  context.init(keyManagers, trustManagers, null);  SSLSocketFactory factory = context.getSocketFactory();  } catch (KeyStoreException e) {  e.printStackTrace();  } catch (NoSuchAlgorithmException e) {  e.printStackTrace();  } catch (CertificateException e) {  e.printStackTrace();  } catch (IOException e) {  e.printStackTrace();  } catch (UnrecoverableKeyException e) {  e.printStackTrace();  } catch (KeyManagementException e) {  e.printStackTrace();  } |

1. Submit WS-Trust RST

|  |
| --- |
| InputStream is = null;  BufferedReader br = null;  String line = null;  StringBuffer result = new StringBuffer();  HttpsURLConnection connection = null;  try {  //SET TO WSTRUST\_IDP\_URL OR WSTRUST\_SP\_URL  String WSTRUST\_URL = getWsTrustURL();    //SET TO WSTRUST\_IDP\_URL OR WSTRUST\_SP\_URL  URL url = new URL(WSTRUST\_URL);  connection = (HttpsURLConnection) url.openConnection();  if (connection instanceof HttpsURLConnection) {  ((HttpsURLConnection) connection).setSSLSocketFactory(sslSocketFactory);  }  //PROCESS RST TO SET TIMESTAMPS & IDs & TOKEN  //SMSESSION value or OpenToken value or SAML 2.0 Assertion  String token = getToken();  String RST = customizeRST(RST\_REQUEST, token);  //Submit RST  connection.setRequestMethod("POST");  connection.setRequestProperty("SOAPAction", "\"\"");  connection.setRequestProperty("Content-type", "text/xml; charset=utf-8");  connection.setRequestProperty("Content-Length", ""+RST.length());  connection.setReadTimeout(CONNECTION\_READ\_TIMEOUT\*1000);  connection.setAllowUserInteraction(false);  connection.setDoInput(true);  connection.setDoOutput(true);  connection.setFollowRedirects(false);  OutputStream out = connection.getOutputStream();  String outStreamCharSet = "UTF-8";    OutputStreamWriter wout = new OutputStreamWriter(out,outStreamCharSet);  wout.write(RST);  wout.flush();  out.close();    is = connection.getInputStream();  br = new BufferedReader(new InputStreamReader(is));  while((line = br.readLine()) != null) {  result.append(line);  }  } catch (SocketTimeoutException ste) {  System.out.println("Socket Timedout, hence the exception");  ste.printStackTrace();  if (connection != null) {  try {  connection.disconnect();  } catch (Exception ste2) {  ste2.printStackTrace();  }  connection = null;  }  result = new StringBuffer("outage");  } catch (Exception e) {  e.printStackTrace();  result = new StringBuffer("exception");  } finally {  try {  if (br !=null) {  br.close();  br = null;  }  } catch (Exception e) {}  }  //PARSE RESPONSE RSTR TO GET OUTPUT TOKEN |