Cisco OAuth Integration Guide for CSP

COPS -Security Services

Cisco IT GIS COPS Security Services Team (asp-web-security@cisco.com)

Author: Sean Zhang (xuexzhan@cisco.com), Aakash Wasnik (awasnik@cisco.com)

Last Edited: 2/27/2015 10:45:19 AM

This document contains the necessary information CSP to integrate with Cisco OAuth Infrastructure. URL provided in the examples is based on the Cisco Non-Production Environment.

Table of Contents

1 *Introduction*

1.1 How to Use This Document

This document contains the necessary information ASP to integrate with Cisco OAuth Infrastructure. URL provided in the examples is based on the Cisco Non-Production Environment.

1.2 What is OAuth?

OAuth is an open-source specification for building a framework for allowing a third-party app (the "client") to access protected resources from another application (the "provider," or "resource owner") at the request of a "user" of the client app. OAuth allows the user to enter his user credentials (ex. username and password) only to the provider app, which then grants the client app permission to view the protected resources on behalf of the user.

1.3 What is OpenID Connect?

OpenID Connect 1.0 is a simple identity layer on top of the OAuth 2.0 protocol. It allows Clients to verify the identity of the End-User based on the authentication performed by an Authorization Server, as well as to obtain basic profile information about the End-User in an interoperable and REST-like manner.

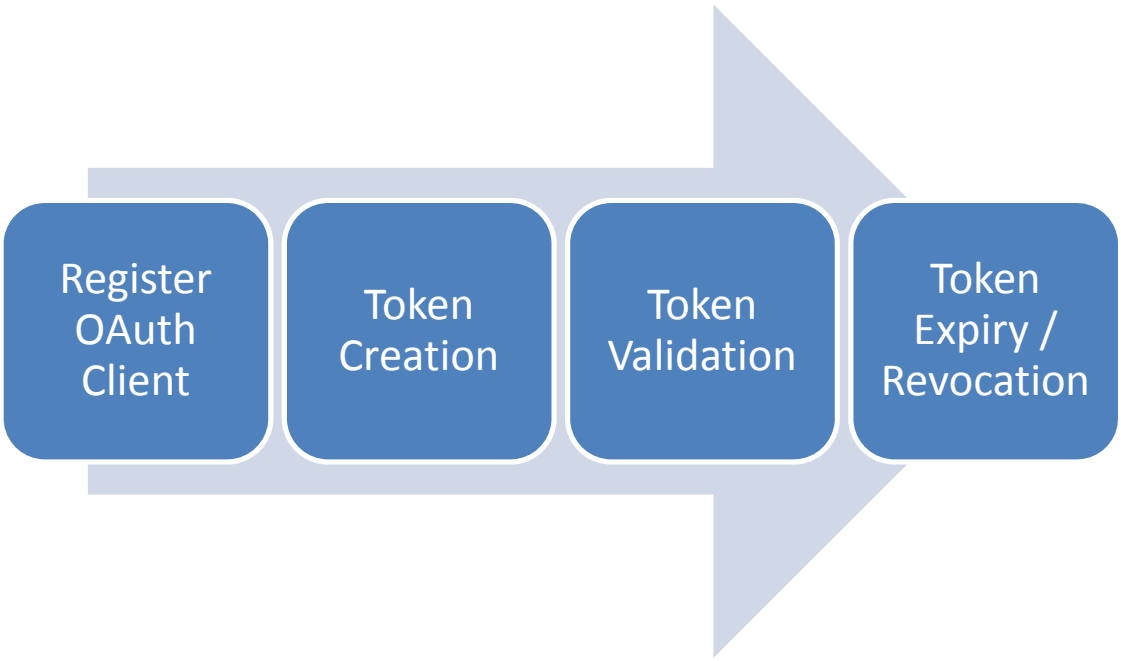
OpenID Connect allows clients of all types, including Web-based, mobile, and JavaScript clients, to request and receive information about authenticated sessions and end-users. The specification suite is extensible, allowing participants to use optional features such as encryption of identity data, discovery of OpenID Providers, and session management, when it makes sense for them.

1.4 Common Terms

- ***Resource Owner*** – An entity capable of granting access to a protected resource. When the resource owner is a person, it is referred to as an end-user.
- ***Resource Server*** - The server hosting the protected resources, capable of accepting and responding to protected resource requests using access tokens.
- ***Client*** – An application making protected resource requests on behalf of the resource owner and with its authorization. The term client does not imply any particular implementation characteristics (e.g. whether the application executes on a server, a desktop, or other devices).
- ***Authorization Server*** - The server issuing access tokens to the client after successfully authenticating the resource owner and obtaining authorization.
- ***Client id/Client secret*** – Credentials provided to Client to make secure interactions with Authorization Server.

- ***Access Token*** – It is a short lived token generated on behalf of resource owner. This token is form of resource owner's authorization to access protected data.
- ***Refresh Token*** – It is a long lived token. Once Access Token is expired, this token can be exchanged with authorization server to get new pair of access token & refresh token.

1.5 OAuth Life Cycle



1.6 Token Timeout / Expiry

There are two tokens generated in OAuth flow. One is Access Token (Short lived Token) & other one is Refresh Token (Long lived Token). By default **Access Token** is valid for **60 mins** & **Refresh Token** is valid for **30 days**. Please note Refresh Token is generated only if "Refresh Token" grant type is set in conjunction with either Authorization Code grant type or Resource Owner Credentials grant type.

2 OAuth Client Management APIs

2.1 Overview

Cisco-PingFederate provides REST based web services to manage OAuth Clients. These REST web services can be accessed by authorized user only. REST web services are available on following urls:

Stage: https://cloudsso-test.cisco.com/oauth/clients

Production: https://cloudsso.cisco.com/oauth/clients

The required MIME type is **application/json**

Below mentioned HTTP headers must be sent for each REST web service request:

userid=""

password=""

Authorized Userid provided here must be a member of "pf-oauth-api" group. Please contact asp-web-security team (asp-web-security@cisco.com) to get more details about how to request access to these REST web services.

2.2 Register/ Create new OAuth Client

HTTP POST method executed on REST Web Services URL creates a new client based on the JSON parameters sent in the request.

2.2.1 Request

Please see below table for description of each request parameter. The required MIME type is **application/json**

Parameter Name	Parameter Description
clientId	(**Required**) A unique ID for the client.
name	(**Required**) A descriptive name for the client.
secret	Client password or phrase. ***Required*** *for the Client Credentials and Access Token Validation grant types only.*
description	A description of what the client application does, displayed in browser when the user is prompted for authorization.

redirectUris	The URI(s) to which the OAuth AS redirects the resource owner's user agent after authorization is obtained. Multiple redirection uri can be specified. ***Required*** *for Implicit and Authorization Code grant types only.*
grantTypes	(**Required**) One or more grant types allowed for the client. Allowed Values: • authorization_code • password (*Resource Owner Password Credentials*) • refresh_token (*Use with authorization_code and/or password grant types*.) • client_credentials • implicit • extension (*SAML 2.0 Bearer*) • urn:pingidentity.com:oauth2:grant_type:validate_ bearer (*Access Token Validation*) **Notes:** • At least one grant type is required. • Separate multiple values with commas. • Access Token Validation cannot be used with any other grant type.
refreshRolling	Indicates whether a new refresh token is issued with each new access token. Allowed values: **true** or **false**. When not provided, the global setting for the Authorization Server is used
persistentGrantExpirationType	Indicates whether to override the global setting for the AS. **Allowed values:** • SERVER_DEFAULT (the default) – Use the global setting for the AS. • NONE – Grants do not expire, regardless of the global setting. • OVERRIDE_SERVER_DEFAULT – Use with both of the persistentGrant* parameters below to set the expiration time period.
persistentGrantExpirationTime	An integer representing units of time for storage of persistent grants for this client—use with persistentGrantExpirationTimeUnit (see below) and only

	when persistentGrantExpirationType is set to OVERRIDE_SERVER_DEFAULT.
persistentGrantExpirationTimeUnit	Units for the expiration time set in the parameter above (if applicable). Allowed values: • h – hours • d – days
bypassApprovalPage	If set to **true**, user consent to resource access is assumed and the approval page is not presented.
restrictScopes	If set to true, limits client access to a subset of the scopes defined for the AS (see Authorization Server Settings). Scopes are limited to the default scope and any listed for restrictedScopes (see below). If no scopes are listed and this parameter is true, only the default scope is available for the client.
restrictedScopes	When used with restrictScopes, limits access to the scope(s) provided in the JSON list, in addition to the default scope. Scopes specified here must be defined manually via admin console. Please check with asp-web-security-team (asp-web-security@cisco.com) about valid values for scopes.
logoUrl	A URL for the logo presented to the user on the grant revocation page.

2.2.2 Required / Optional Parameters per Grant Type

Please see below table for required & optional parameter per Grant Type

Grant Type	Required Parameter	**Optional Parameter**
authorization_code	• clientId • name • grantTypes • redirectUris • secret	• description • refreshRolling • persistentGrantExpirationType • persistentGrantExpirationTime

		• persistentGrantExpirationTimeUnit • bypassApprovalPage • restrictScopes • restrictedScopes • logoUrl
implicit	• clientId • name • grantTypes • redirectUris	• secret • description • refreshRolling • persistentGrantExpirationType • persistentGrantExpirationTime • persistentGrantExpirationTimeUnit • bypassApprovalPage • restrictScopes • restrictedScopes • logoUrl
password (Resource Owner Password Credentials)	• clientId • name • grantTypes • secret	• redirectUris • description • refreshRolling • persistentGrantExpirationType • persistentGrantExpirationTime • persistentGrantExpirationTimeUnit • bypassApprovalPage • restrictScopes • restrictedScopes • logoUrl
client_credentials	• clientId • name • grantTypes • secret	• redirectUris • description • refreshRolling • persistentGrantExpirationType

		• persistentGrantExpirationTime • persistentGrantExpirationTimeUnit • bypassApprovalPage • restrictScopes • restrictedScopes • logoUrl
refresh_token (Use with authorization_code and/or password grant types.)	• clientId • name • grantTypes • secret	• redirectUris • description • refreshRolling • persistentGrantExpirationType • persistentGrantExpirationTime • persistentGrantExpirationTimeUnit • bypassApprovalPage • restrictScopes • restrictedScopes • logoUrl
urn:pingidentity.com:oauth2:grant_type:validate_bearer (Access Token Validation)	• clientId • name • grantTypes • secret	• redirectUris • description • refreshRolling • persistentGrantExpirationType • persistentGrantExpirationTime • persistentGrantExpirationTimeUnit • bypassApprovalPage • restrictScopes • restrictedScopes • logoUrl

2.2.3 Response

HTTP Status Code	Reason	Description
200	Success	OAuth Client created successfully.
400	Failed To Create Client	The response contains details as to why the client creation failed.
500	Internal Server Error	An unknown error has occurred.

2.2.4 Example

URL: https://cloudsso-test.cisco.com/oauth/clients

HTTP Method: POST

MIME Type: application/json

JSON Request:

{"client":[{"clientId":"sampleclient","name":"sampleclient","refreshRolling":"true",
"redirectUris":["http://www.example.com", http://www.example2.com"],
"logoUrl":"http://www.example.com/logo.gif","secret":"thatssecret",
"description":"Description of the sampleclient","persistentGrantExpirationType":"OVERRIDE_SERVER_DEFAULT","persistentGrantExpirationTime":"3","persistentGrantExpirationTimeUnit":"d","bypassApprovalPage":"true",
"restrictScopes":"true","restrictedScopes":
["Read", "Write"],
"grantTypes":["password","refresh_token"]}]}

Response: 200 OK

2.3 Read existing OAuth Client

HTTP GET method executed on REST Web Services URL (/oauth/clients/<oauth client id>) retrieves details of specified OAuth client.

2.3.1 Request

Add OAuth Client identifier to the REST Web Service URL.

/oauth/clients/<oauth client id>

2.3.2 Response

HTTP Status Code	Reason	Description
200	Success	JSON client parameters are included. **Note: The parameter refreshRolling is not returned if the AS global setting is set for a client (the default).**
400	Failed To Retrieve Client	The response contains details as to why client could not be retrieved.
500	Internal Server Error	An unknown error has occurred.

2.3.3 Example

Assume "sampleclient" is the client identifier of the existing OAuth Client

URL: https://cloudsso-test.cisco.com/oauth/clients/sampleclient

HTTP Method: GET

Request: No parameter

Response:

{"bypassApprovalPage":"true","clientId":"sampleclient","description":"Description of the sampleclient","grantTypes":["refresh_token","password"],"logoUrl":"http://www.example.com/logo.gif","name":"sampleclient","persistentGrantExpirationTime":"3","persistentGrantExpirationTimeUnit":"d","persistentGrantExpirationType":"OVERRIDE_SERVER_DEFAULT","redirectUris":["http://www.example.com","http://www.example2.com"],"refreshRolling":"true","restrictScopes":"true","restrictedScopes":["Read","Write"]}

2.4 Update / Modify existing OAuth Client

HTTP PUT method executed on REST Web Services URL updates existing client based on the JSON parameters sent in the request. Please note the Client Identifier (clientId) cannot be modified via this method.

2.4.1 Request

The same parameters described for POST apply for PUT with one addition. The required MIME type is **application/json**

Parameter Name	Parameter Description
forceSecretChange	Use this parameter, set to true, in conjunction with the secret parameter to change a client pass phrase. **NOTE:** If the secret parameter is used without forceSecretChange, the secret value is ignored.

2.4.2 Response

HTTP Status Code	Reason	Description
200	Success	The body contains a list of updated client.
400	Failed To Update Client	The response contains details as to why the client could not be updated.
500	Internal Server Error	An unknown error has occurred.

2.4.3 Example

URL: https://cloudsso-test.cisco.com/oauth/clients

HTTP Method: PUT

MIME Type: application/json

JSON Request:

{"client":[{"clientId":"12345","name":"ClientDoe","refreshRolling":"true",
redirectUris": ["http://www.url.com", http://www.url2.com"]",
logoUrl":"http://www.url.com/image.gif",
"forceSecretChange":"true","secret":"mysecretphrase","description":"Description","persistentGrantExpirationType":"OVERRIDE_SERVER_DEFAULT","persistentGrantExpirationTime":"3","persistentGrantExpirationTimeUnit":"d",bypassApprovalPage":"true","restrictScopes":"true", "restrictedScopes":["scope 1","scope 2"]"grantTypes":["password","refresh_token"]}]}

Response:

{"client":{"bypassApprovalPage":"true","clientId":"sampleclient","description":"Description of the sampleclient","grantTypes":["refresh_token","password"],"logoUrl":"http://www.example.com/logo.gif","name":"sampleclient","persistentGrantExpirationTime":"3","persistentGrantExpirationTimeUnit":"d","persistentGrantExpirationType":"OVERRIDE_SERVER_DEFAULT","redirectUris":["http://www.example.com","http://www.example2.com"],"refreshRolling":"true","restrictScopes":"true","restrictedScopes":["Read","Write"]}}

2.5 Delete existing OAuth Client

HTTP DELETE method executed on REST Web Services URL (/oauth/clients/<oauth client id>) deletes record of specified OAuth client.

2.5.1 Request

Add OAuth Client identifier to the REST Web Service URL.

/oauth/clients/<oauth client id>

2.5.2 Response

HTTP Status Code	Re**ason**	**Description**
200	Success	OAuth Client deleted Successfully.
400	Failed To Delete Client	The response contains details as to why client could not be deleted
405	Method Not Allowed	The client ID was not specified.
500	Internal Server Error	An unknown error has occurred.

2.5.3 Example

URL: https://cloudsso-test.cisco.com/oauth/clients/sampleclient

HTTP Method: DELETE

Request: No parameter

Response: HTTP 200

3 OAuth Client Registration (Legacy Web Service)

3.1 OAuth Client Creation URL

OAuth Client Creation URL:

Stage: https://cloudsso-test2.cisco.com/clientoauth/create

Producation: https://cloudsso2.cisco.com/clientoauth/create

Only HTTP POST operation is allowed

HTTP header must to be sent for each request to OAuth Client Creation URL:

userid=""

password=""

Userid provided here must be a member of "pf-oauth-api" group.

Required Parameter for each request sent to OAuth Client Creation URL:

ws_id="#ws_id#"

ws_pw="#ws_pw#"

There are other parameters that are expected for the OAuth Client Creation API.

Parameter Name	Parameter Description	Value
client_id	Client Identifier	Any Text
client_pw	Client Secret	Any Text
client_name	Client Name	Any Text
client_desc	Client Description	Any Text
redirect_uri	Redirection URL	URL format
logo_uri	Logo URL	URL format
grant_type_authorization_code	Authorization Code grant type	"true"
grant_type_password	Resource owner password credentials grant type	"true"
grant_type_refresh_token	Refresh Token grant type	"true"
grant_type_implicit	Implicit grant type	"true"

grant_type_client_credentials	Client Credentials grant type	"true"
grant_type_accesstoken_validation	Access Token Validation grant type	"true"

3.2 OAuth Client Creation for Different Grant Types

There are two important condition regarding refresh token and Access Token validation grant type.

1. The Refresh Token grant type needs to be used in conjunction with the either the Authorization Code or Resource Owner Password Credentials grant types
2. Access token validation grant type cannot be set in conjunction with other grant types.

Grant Type	**Authorization Code**
Required Parameter	client_id,client_name,redirect_uri, grant_type_authorization_code
Optional Parameter	client_desc, logo_uri, client_pw
Sample Data	ws_id=<ws_id> ws_pw=<ws_pw> client_id=cl12345 client_name=cl12345 redirect_uri=https://www.example.com/redirection grant_type_authorization_code =true

Grant Type	**Resource Owner Password Credentials**
Required Parameter	client_id , client_name, grant_type_password
Optional Parameter	client_desc, logo_uri, redirect_uri , client_pw
Sample Data	ws_id=<ws_id> ws_pw=<ws_pw> client_id=cl12345 client_name=cl12345 grant_type_password=true

Grant Type	**Implicit**
Required Parameter	client_id , client_name,redirect_uri, grant_type_implicit
Optional Parameter	client_desc, logo_uri, client_pw
Sample Data	ws_id=<ws_id> ws_pw=<ws_pw> client_id=cl12345 client_name=cl12345 redirect_uri=https://www.example.com/redirection grant_type_implicit=true

Grant Type	Refresh Token
Required Parameter	client_id , client_name, grant_type_refresh_token
Optional Parameter	client_desc, logo_uri,redirect_uri, client_pw
Sample Data	ws_id=<ws_id> ws_pw=<ws_pw> client_id=cl12345 client_name=cl12345 grant_type_refresh_token=true grant_type_authorization_code=true

Grant Type	Client Credentials
Required Parameter	client_id , client_name, client_pw, grant_type_client_credentials
Optional Parameter	client_desc, logo_uri, redirect_uri
Sample Data	ws_id=<ws_id> ws_pw=<ws_pw> client_id=cl12345 client_pw=clientpassword client_name=cl12345 grant_type_client_credentials=true

Grant Type	Access Token Validation
Required Parameter	client_id,client_name,client_pw, grant_type_accesstoken_validation
Optional Parameter	client_desc, logo_uri, redirect_uri
Sample Data	ws_id=<ws_id> ws_pw=<ws_pw> client_id=cl12345 client_pw=clpwd client_name=cl12345 grant_type_accesstoken_validation=true

3.3 OAuth Client Creation Response

After POST to Client Creation URL, the servers will respond with a HTTP status code.

HTTP Status Code	Reason	Description

200	OAuth Client creation successful	OAuth Client creation successful
400	Data Validation error	Required parameter missing for any grant type OR OAuth client with specified id already present.
401	Web Service Authentication Failed	ws_id & ws_pw not present or has incorrect values
500	Internal Server Error	Any error occurred at server while processing request

4 Token Creation

4.1 Client Identification and Authentication

Clients can authenticate to the OAuth AS using their client identifier and client secret in two ways

- **HTTP Basic authentication scheme** (where the client identifier is the username, and the client secret is the password) - **Recommended**
- **HTTP request parameters** (client_id and client_secret)

Clients without a client secret can use the client_id parameter to identify themselves to the OAuth AS and omit the client_secret parameter.

4.2 Grant Type: Authorization Code

Generating Access Token / Refresh token via Authorization Code grant type is a two-step process. First step is to generate the Authorization Code & Second step is to exchange authorization code for Access Token / Refresh Token.

4.2.1 Generate Authorization Code

- **Authorization Endpoint URL**:
  Stage: https://cloudsso-test.cisco.com/as/authorization.oauth2
  Production: https://cloudsso.cisco.com/as/authorization.oauth2
- **Note:**
  - If redirect uri is not specified in the request then authorization code is sent on the redirect uri registered for OAuth Client in OAuth Authorization Server.
  - Authorization code is valid only for 60 seconds.

	Generate Authorization Code
Required Parameter	client_id, response_type
Optional Parameter	state, scope, redirect_uri

- **Please see below description of each parameter.**

Parameter Name	**Description**
client_id	**(Required)** The client identifier
response_type	**(Required)** A value of **code** results in the Authorization Code grant type

	response_type=code
redirect_uri	Optional for clients with only one specific redirect URI configured. Required if more than one URI is configured in PingFederate for the client or if a wildcard is used for a single URI entry.
state	An opaque value used by the client to maintain state between the request and callback. If included, the AS returns this parameter and the given value when redirecting the user agent back to the client.
scope	The scope of the access request expressed as a list of space-delimited, case-sensitive strings. Valid scope values are defined on the OAuth AS settings screen. Please check with asp-web-security (asp-web-security@cisco.com) team to get information about valid scope values.

- **Flow Diagram:** Please see section 8.1 for flow diagram.

- **Example:**

HTTP Method: GET
Request:
https://cloudsso-test.cisco.com/as/authorization.oauth2?response_type=code
&client_id=<clientid>
&state=<state>
&redirect_uri=http://client_redirecturl
&scope=<scopevalue>
Response:
http://client_redirecturl
?state=<state>&code=HY5h9MBcgekPGFeCQijnD8ILtEy0UHGOidny1ABm

4.2.2 *Exchange Authorization Code for Access Token / Refresh Token*

- **Token Endpoint URL:**
  Stage: https://cloudsso-test.cisco.com/as/token.oauth2
  Production: https://cloudsso.cisco.com/as/token.oauth2
- **Note:**
  - HTTP POST method should be used to pass the required parameters.
  - Required parameters must be passed in request-body.
  - Refresh Token is only generated if "Refresh Token" grant type is configured for OAuth Client in conjunction with "Authorization Code" grant type .

	Exchange Authorization Code for AT/RT

Required Parameter	client_id, client_secret, code, grant_type, redirect_uri

- **Please see below description of each parameter.**

Parameter Name	Description
client_id	**(Required)** The client identifier
client_secret	**(Required)** The client secret
redirect_uri	**(Required)** The value here must match the url specified while generating the authorization code in first step.
code	**(Required)** The authorization code received from the authorization server in first step.
grant_type	**(Required)** "authorization_code" must be passed as value of this parameter.

- **Flow Diagram:** Please see section 8.1 for flow diagram.

- **Example:**

HTTP Method: POST
Request:
https://cloudsso-test.cisco.com/as/token.oauth2
(POST Parameter in request body)
client_id : <clientid>
client_secret : <clientsecret>
grant_type : authorization_code
redirect_uri : https://client_redirecturl
code : b_HkJNHQ8BZ-qjUpDmmqDuUorvqsm6BMzifqDGqD
Response:
{"token_type":"Bearer","refresh_token":"OEGnqiHxdSiUSTNKbiyH7s70T0LMA6EtwdfLnRoylR","
access_token":"ov2dv0PVXtdOqWVziMTHuWAzbZHP"}

4.3 Grant Type: Resource Owner Credentials

- **Token Endpoint URL:**
  Stage: https://cloudsso-test.cisco.com/as/token.oauth2
  Production: https://cloudsso.cisco.com/as/token.oauth2
- **Note:**
  - HTTP POST method should be used to pass the required parameters.
  - Required parameters must be passed in request-body.
  - Refresh Token is only generated if "Refresh Token" grant type is configured for OAuth Client in conjunction with "Resource Owner Credentials" grant type.

	Resource Owner Credentials Grant Type
Required Parameter	client_id, client_secret, grant_type, username, password
Optional Parameter	scope

- **Please see below description of each parameter.**

Parameter Name	Description
client_id	**(Required)** The client identifier
client_secret	**(Required)** The client secret
username	**(Required)** Resource owner's username. Encoded as UTF-8.
password	**(Required)** Resource owner's password. Encoded as UTF-8.
grant_type	**(Required)** "password" must be passed as a value of this parameter.
scope	The scope of the access request expressed as a list of space-delimited, case-sensitive strings. Valid scope values are defined on the OAuth AS settings screen. Please check with asp-web-security (asp-web-security@cisco.com) team to get information about valid scope values.

- **Flow Diagram:** Please see section 8.3 for flow diagram.

- **Example:**

HTTP Method: POST
Request:
https://cloudsso-test.cisco.com/as/token.oauth2
(POST Parameter in request body)
client_id : <clientid>
client_secret : <clientsecret>
grant_type : password
username : <userid>
password : <userpassword>
Response:
{"token_type":"Bearer","refresh_token":"OEGnqiHxdSiUSTNKbiyH7s70T0LMA6EtwdfLnRoylR","
access_token":"ov2dv0PVXtdOqWVziMTHuWAzbZHP"}

4.4 Grant Type: Implicit

- **Authorization Endpoint URL:**
  Stage: https://cloudsso-test.cisco.com/as/authorization.oauth2
  Production: https://cloudsso.cisco.com/as/authorization.oauth2
- **Note:**

- Implicit grant type is designed for client side applications / code (Javascripts) to use access tokens
- If redirect uri is not specified in the request then authorization code is sent on the redirect uri registered for OAuth Client in OAuth Authorization Server.
- Access Token return on redirect url is preceded by "#" character. Browser understands this special character & makes sure that access token is not passed to the server side.

	Implicit Grant Type
Required Parameter	client_id, response_type
Optional Parameter	state, scope, redirect_uri

- **Please see below description of each parameter.**

Parameter Name	**Description**
client_id	**(Required)** The client identifier
response_type	**(Required)** A value of **token** results in the Implicit grant type *response_type=token*
redirect_uri	Optional for clients with only one specific redirect URI configured. Required if more than one URI is configured in PingFederate for the client or if a wildcard is used for a single URI entry.
state	An opaque value used by the client to maintain state between the request and callback. If included, the AS returns this parameter and the given value when redirecting the user agent back to the client.
scope	The scope of the access request expressed as a list of space-delimited, case-sensitive strings. Valid scope values are defined on the OAuth AS settings screen. Please check with asp-web-security (asp-web-security@cisco.com) team to get information about valid scope values.

- **Flow Diagram:** Please see section 8.2 for flow diagram.

- **Example:**

HTTP Method: GET
Request:
https://cloudsso-test.cisco.com/as/authorization.oauth2?response_type=token

&client_id=<clientid>
&state=<state>
&redirect_uri=http://client_redirecturl
&scope=<scopevalue>
Response:
http://client_redirecturl
#state=<state>&token_type=Bearer&access_token=UfAoH9Wr94XhGkvgZqCH9x6IMJKp

4.5 Grant Type: Client Credentials

- **Token Endpoint URL:**

  Stage: https://cloudsso-test.cisco.com/as/token.oauth2

  Production: https://cloudsso.cisco.com/as/token.oauth2

- **Note:**
  - HTTP POST method should be used to pass the required parameters.
  - Required parameters must be passed in request-body.
  - Refresh token is NOT generated for this grant type. Only Access Token is generated.

	Client Credentials Grant Type
Required Parameter	client_id, client_secret, grant_type
Optional Parameter	scope

- **Please see below description of each parameter.**

Parameter Name	**Description**
client_id	**(Required)** The client identifier
client_secret	**(Required)** The client secret
grant_type	**(Required)** "client_credentials" must be passed as a value of this parameter.
scope	The scope of the access request expressed as a list of space-delimited, case-sensitive strings. Valid scope values are defined on the OAuth AS settings screen. Please check with asp-web-security (asp-web-security@cisco.com) team to get information about valid scope values.

- **Flow Diagram:** Please see section 8.4 for flow diagram.

- **Example:**

HTTP Method: POST

Request:
https://cloudsso-test.cisco.com/as/token.oauth2
(POST Parameter in request body)
client_id : <clientid>
client_secret : <clientsecret>
grant_type : client_credentials
Response:
{"token_type":"Bearer","access_token":"ov2dv0PVXtdOqWVziMTHuWAzbZHP"}

4.6 Grant Type: Refresh Token

- **Token Endpoint URL:**
  Stage: https://cloudsso-test.cisco.com/as/token.oauth2
  Production: https://cloudsso.cisco.com/as/token.oauth2
- **Note:**
  - HTTP POST method should be used to pass the required parameters.
  - Required parameters must be passed in request-body.
  - If Access Token is expired, Refresh Token will be used to get the new Access Token without requesting end user / resource owner to authenticate again.
  - "Refresh Token" grant type can only be used in conjunction with "Authorization Code" & "Resource Owner Password Credentials" grant type.

	Refresh Token Grant Type
Required Parameter	client_id, client_secret, grant_type, refresh_token
Optional Parameter	scope

- **Please see below description of each parameter.**

Parameter Name	**Description**
client_id	**(Required)** The client identifier
client_secret	**(Required)** The client secret
grant_type	**(Required)** "refresh_token" must be passed as a value of this parameter.
refresh_token	**(Required)** The refresh token issued to the client during a previous access-token request.
scope	The scope of the access request expressed as a list of space-delimited, case-sensitive strings. Valid scope values are defined on the OAuth AS settings screen. Please check with asp-web-security (asp-web-security@cisco.com) team to get information about valid scope values.

- **Example:**

HTTP Method: POST
Request:
https://cloudsso-test.cisco.com/as/token.oauth2
(POST Parameter in request body)
client_id : <clientid>
client_secret : <clientsecret>
grant_type : refresh_token
refresh_token : GESnqiHxdSiUSTNKbiyH7s70T0LPQ6EtwdfLnPOdFr
Response:
{"token_type":"Bearer","refresh_token":"OEGnqiHxdSiUSTNKbiyH7s70T0LMA6EtwdfLnRoylR","
access_token":"ov2dv0PVXtdOqWVziMTHuWAzbZHP"}

5 Token Validation

5.1 Grant Type: Access Token Validation

- **Token Endpoint URL:**

  Stage: https://cloudsso-test.cisco.com/as/token.oauth2

  Production: https://cloudsso.cisco.com/as/token.oauth2

- **Note:**
  - The grant type allows an Resource Server to check with the OAuth Authorization Server on the validity of a bearer access token that it has received from a client making a protected-resources call.
  - OAuth Client (client_id / client_secret) is created with grant type "urn:pingidentity.com:oauth2:grant_type:validate_bearer". This OAuth Client represents Resource Server.
  - HTTP POST method should be used to pass the required parameters.
  - Required parameters must be passed in request-body.

	Access Token Validation Grant Type
Required Parameter	client_id, client_secret, grant_type, token

- **Please see below description of each parameter.**

Parameter Name	**Description**
client_id	**(Required)** The client identifier
client_secret	**(Required)** The client secret
grant_type	**(Required)** "urn:pingidentity.com:oauth2:grant_type:validate_bearer" must be passed as a value of this parameter.
token	**(Required)** The bearer access token to be validated.

Example:
HTTP Method: POST
Request:
https://cloudsso-test.cisco.com/as/token.oauth2
(POST Parameters)
grant_type=urn:pingidentity.com:oauth2:grant_type:validate_bearer
token=fop3gJf63ihL30F6NGMNc1XDtqmI
client_id=<clientid>
client_secret=<clientsecret>

Response:

{"scope":"Read/Write","token_type":"urn:pingidentity.com:oauth2:validated_token","expires_in":1547,"client_id":"<clientid>","access_token":{"uid":"userid"}}

6 Token Revocation

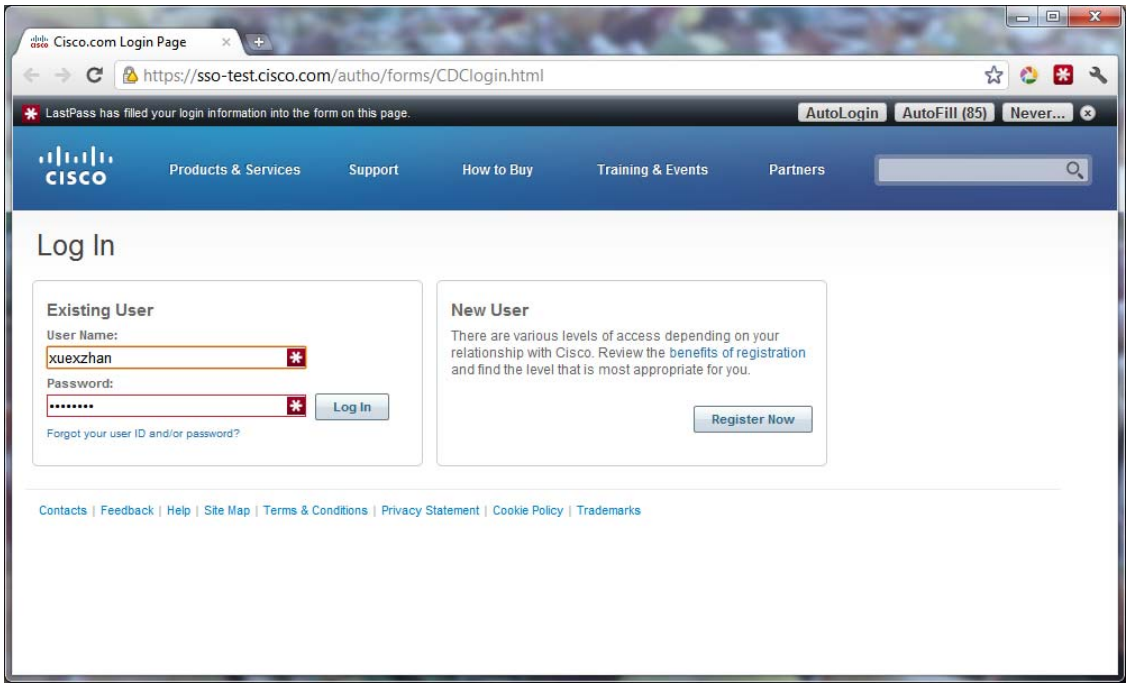
6.1 OAuth Grant Management

Stage: https://cloudsso-test.cisco.com/as/oauth_access_grants.ping

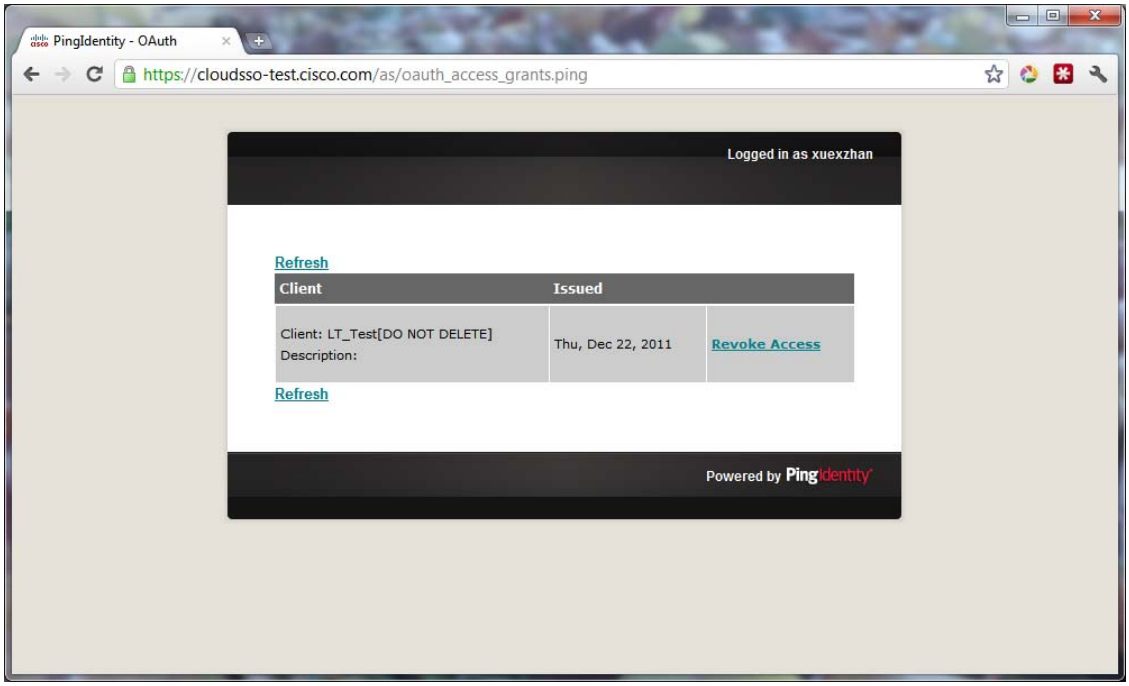
Production: https://cloudsso.cisco.com/as/oauth_access_grants.ping

The grants endpoint is where end-users/resource owners go to view (and optionally revoke) the persistent access grants they have made. This endpoint is not part of the OAuth specification, but many OAuth providers offer a similar type of functionally. The grants displayed are those associated with the USER_KEY of the authenticated user. The same attribute mapping(s) from the authentication source to USER_KEY used for the authorization endpoint are used here to look up the user's existing grants.

Having the user login using their CDC Credentials



A list of granted Client Access will be displayed and the user may click on "Revoke Access" if needed.


PingIdentity - OAuth
https://cloudsso-test.cisco.com/as/oauth_access_grants.ping
Logged in as xuexzhan
Refresh
Client
Issued
Client: LT_Test[DO NOT DELETE]
Description:
Thu, Dec 22, 2011
Revoke Access
Refresh
Powered by PingIdentity

7 OpenID Connect

7.1 OpenID Connect Request

OpenID Connect could be used with the following grant types: Authorization Code, Resource Owner Credentials, and Implicit. Client Credentials could not use OpenID Connect because it does not contain any user information. Application will need to include additional scope values when making a request for OpenID connect.

	OpenID Connect Request
Required Parameter	Refer to the Section 4 for the required parameter for each grant type
Additional Required Parameter	Scope

Scope Value	**Description**	**Attribute**
openid	**(Required)** OpenID Connect Default Scope	None
profile	**(Required)** Access to Basic User Information	title access_level company family_name given_name
phone	**(Optional)** Access to Phone Number	phone_number
email	**(Optional)** Access to Primary Email Address	email_verified

- **Flow Diagram:** Please see section 8.5 for flow diagram.

- **Example:** using Authorization Code Grant Type

HTTP Method: GET
Request:
https://cloudsso-test.cisco.com/as/authorization.oauth2?response_type=code
&client_id=<clientid>
&state=<state>
&redirect_uri=http://client_redirecturl
&scope=<scopevalue>%20openid%20profile%20email%20phone
Response:
http://client_redirecturl
?state=<state>&code=HY5h9MBcgekPGFeCQijnD8ILtEy0UHGOidny1ABm

HTTP Method: POST
Request:
https://cloudsso-test.cisco.com/as/token.oauth2
(POST Parameter in request body)
client_id : <clientid>
client_secret : <clientsecret>
grant_type : authorization_code
redirect_uri : https://client_redirecturl
code : HY5h9MBcgekPGFeCQijnD8ILtEy0UHGOidny1ABm
Response:

{"token_type":"Bearer","expires_in":3599,"refresh_token":"NecuOiZ90PCDqHBvZp6sQonDkxon7iEuHYMEXmh7LI","id_token":"eyJhbGciOiJSUzI1NiIsImtpZCI6Ijk2anV6In0.eyJzdWIiOiJ4dWV4emhhbiIsImF1ZCI6IkNsaWVudFNvbGFpTlBSRCIsImp0aSI6Imtjckd4TkpZcG40N2tHdEZrOTcxUzYiLCJpc3MiOiJodHRwczpcL1wvY2xvdWRzc28tdGVzdC5jaXNjby5jb20iLCJpYXQiOjE0MjQ4OTA4NzMsImV4cCI6MTQyNDg5MTE3M30.cF7bQK_vD9MbWQBW-02XvxXZ8Nbo6_gXFlWQmgeUTZbLcCPgon94eVjAayaQwGCcaCeIzpvc3iIJIOaPIHsdmLo7qZUahpkx4jcY-3ZZGcc1wliPUYnscxcmiQ0z1-FySuNtH3rwIPfymC71mcB4ZaGaxM6BgEPsTdi8a9tbdU8y8-kC1TkfH6PR6glus6lmaRhiSHJTM3xuZQbLPxURcWfL2lQ-vS1_aPD7a-Qjr-KRa2U3IvyKOtnc4k5JmZYEDuHhTVmAHfkpba2lIDOhAPFglAddWczXOkYbTTGmR1cC0h63GtMRR0nQNc3rAtoG1S0eSeb2f2dW-ozdybGk3A","access_token":"rndddPFvHXB3lLzYPeRTHIfGgVWi"}

7.2 OpenID Connect User Information Endpoint

Stage: https://cloudsso-test.cisco.com/idp/userinfo.openid

Production: https://cloudsso.cisco.com/idp/userinfo.openid

The user info endpoint is where application exchanges Access Token for authorized user Information.

	OpenID Connect User Information Lookup
Required Parameter	Authorization

- **Please see below description of each parameter.**

Parameter Name	**Description**
Authorization	**(Required)** Access Token validated for the Scope of openid

- **Example:**

HTTP Method: Get
Request:
https://cloudsso-test.cisco.com/idp/userinfo.openid
(GET Parameter in HTTP HEADER)

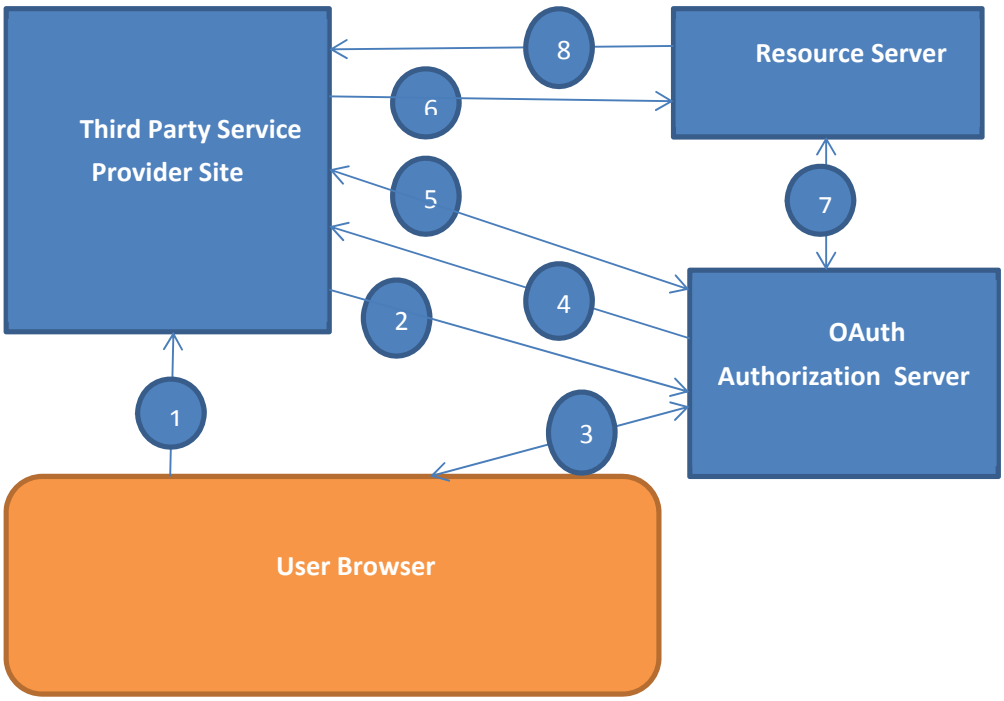
Authorization: Bearer <Access Token Value>

Response:

{"sub":"xuexzhan","title":"ENGINEER.IT","phone_number":"+1 408 424 6835","access_level":"4","email_verified":"xuexzhan@cisco.com","company":"Cisco Systems, Inc.","family_name":"Zhang","given_name":"Xuexin"}

8 Flow Diagram

8.1 Authorization Code Flow



Note:

1. Third party service provider mentioned above could be web application / mobile application / desktop application

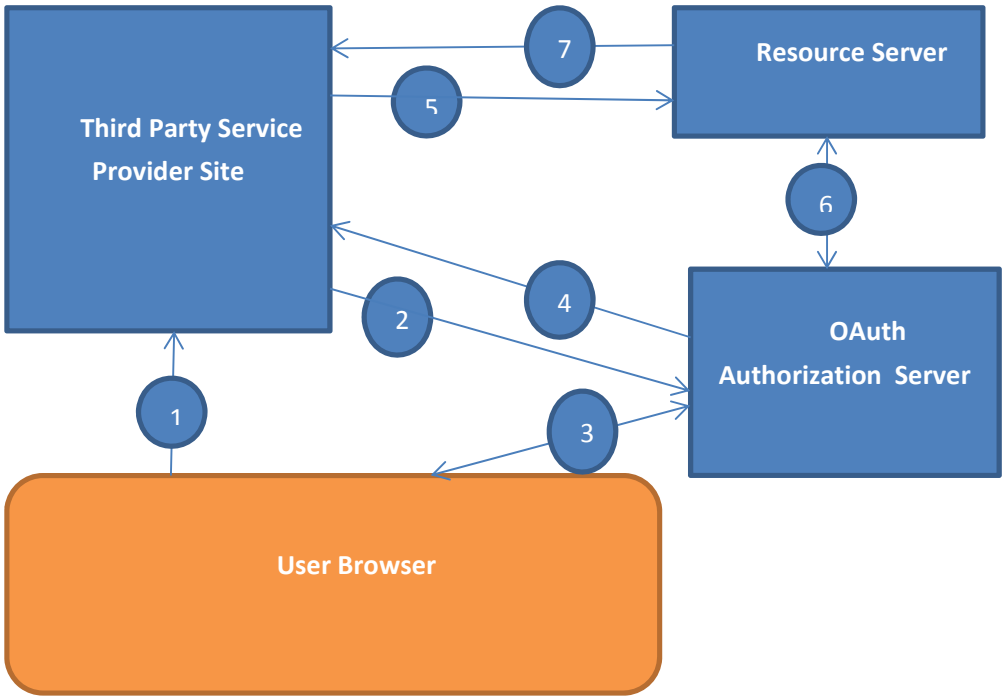
Pre-requisite:

1. Third party service provider is registered with OAuth Authorization server. As a part of the registration client_id & client_secret has been provided to securely interact with OAuth Authorization Server.
2. Resource server is also registered with OAuth Authorization Server. As a part of the registration client_id & client_secret has been provided to securely interact with OAuth Authorization Server.

Flow:

1. Resource owner (end user / real user) accesses the third party service provider site. Third party site requires access to the Resource owner's protected resource (Maintained by separate entity called resource server).
2. Service provider sites redirects user to OAuth Authorization Server to get Resource Owner's consent.
3. OAuth Authorization server would make sure Resource Owner is authenticated and provides consent.
4. If Resource Owner provides consent then OAuth Authorization server generates the Authorization Code and sends it to the service provider site's url registered with OAuth Authorization Sever.
5. Third Party Service provider exchanges the Authorization Code with OAuth Authorization Server to receive Access Token & *Refresh Token (provided pre-requisites for generating refresh tokens are met).
6. Third Party Service provider makes call to Resource Server to access Resource Owner's protected resource & sends access token along with the request.
7. Resource Servers send access token to OAuth Authorization Server to check the validity of the token.
8. Resource Server gets the descriptive information about the token from OAuth Authorization Server (such as resource owner to which token was issued, third party service provider's identification that possesses the access token). Based on this information & additional checks Resource server decides whether to provide access to resource's owner protected contents to third part service provider. Resource server sends protected resource's information in response to Third party service provider.

8.2 Implicit Flow



Note:

1. Third party service provider mentioned above could be web application developed via Client Side programming language such as Java Scripts.

Pre-requisite:

1. Third party service provider is registered with OAuth Authorization server. As a part of the registration client_id has been provided to securely interact with OAuth Authorization Server.

2. Resource server is also registered with OAuth Authorization Server. As a part of the registration client_id & client_secret has been provided to securely interact with OAuth Authorization Server.

Flow:

1. Resource owner (end user / real user) accesses the third party service provider site. Third party site requires access to the Resource owner's protected resource (Maintained by separate entity called resource server).

2. Service provider sites redirects user to OAuth Authorization Server to get Resource Owner's consent.

3. OAuth Authorization server would make sure Resource Owner is authenticated and provides consent.

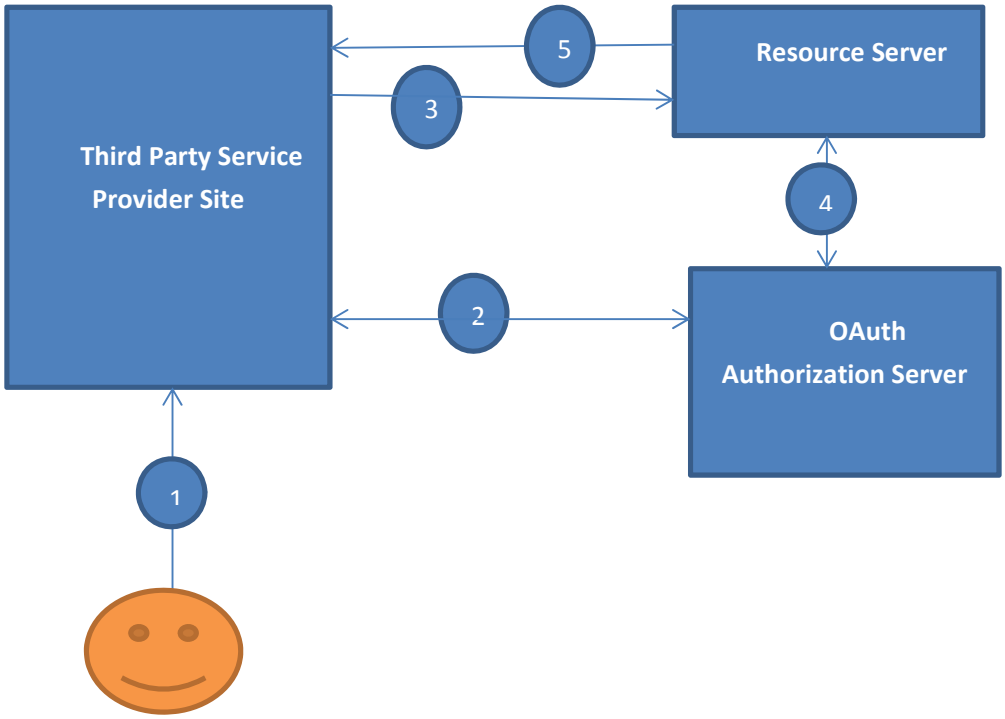
4. If Resource Owner provides consent then OAuth Authorization server generates the Access Token and sends it to the service provider site's url registered with OAuth Authorization Sever. Access Token is sent via query string appended using "#" character so that access token is retrieved only by client side application code such as Javascript.

5. Third Party Service provider makes call to Resource Server to access Resource Owner's protected resource & sends access token along with the request.

6. Resource Servers send access token to OAuth Authorization Server to check the validity of the token.

7. Resource Server gets the descriptive information about the token from OAuth Authorization Server (such as resource owner to which token was issued, third party service provider's identification that possesses the access token). Based on this information & additional checks Resource server decides whether to provide access to resource's owner protected contents to third part service provider. Resource server sends protected resource's information in response to Third party service provider.

8.3 Resource Owner Password Credentials Flow



Note:

1. Third party service provider mentioned above could be mobile application / desktop application.
2. This service provider collect resource owner's credentials using various mechanism (Local login screen)

Pre-requisite:

1. Third party service provider is registered with OAuth Authorization server. As a part of the registration client_id & client_secret has been provided to securely interact with OAuth Authorization Server.

2. Resource server is also registered with OAuth Authorization Server. As a part of the registration client_id & client_secret has been provided to securely interact with OAuth Authorization Server.

Flow:

1. Resource owner (end user / real user) accesses the third party service provider site. Third party site requires access to the Resource owner's protected resource (Maintained by separate entity called resource server). Third party service provider collects resource owner's credentials.

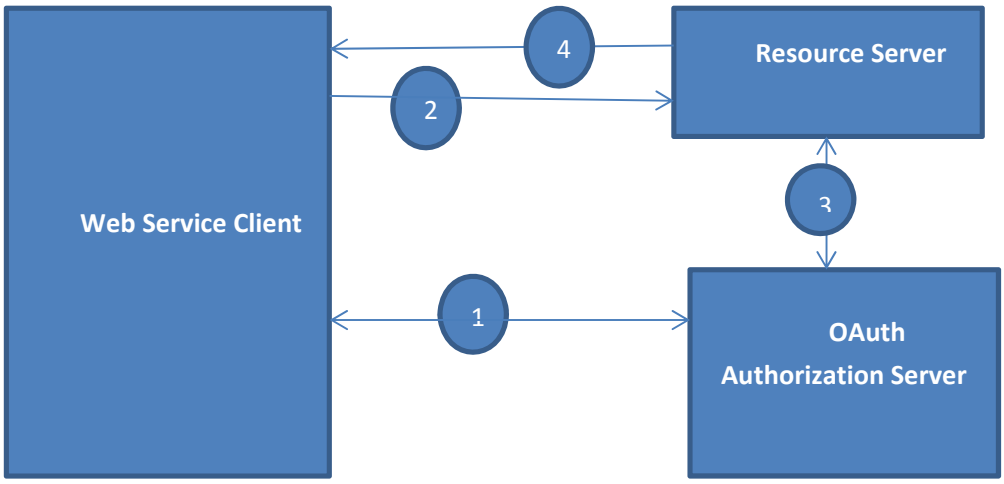
2. Third party Service provider pass Resource owner's credentials & its own credentials along with other required information to OAuth Authorization sever to get access token / refresh token issued on half of resource owner.

3. Third Party Service provider makes call to Resource Server to access Resource Owner's protected resource & sends access token along with the request.

4. Resource Servers send access token to OAuth Authorization Server to check the validity of the token.

5. Resource Server gets the descriptive information about the token from OAuth Authorization Server (such as resource owner to which token was issued, third party service provider's identification that possesses the access token). Based on this information & additional checks Resource server decides whether to provide access to resource's owner protected contents to third party service provider. Resource server sends protected resource's information in response to Third party service provider.

8.4 Client Credentials Flow



Pre-requisite:

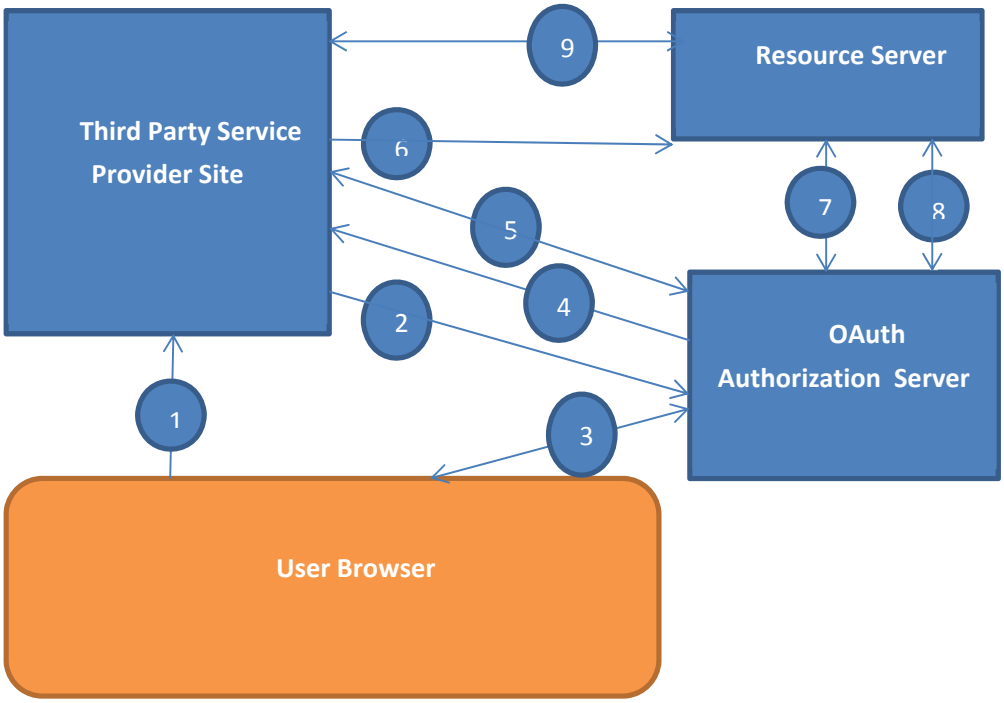
1. Web Service Client is registered with OAuth Authorization server. As a part of the registration client_id & client_secret has been provided to securely interact with OAuth Authorization Server.

2. Resource server is also registered with OAuth Authorization Server. As a part of the registration client_id & client_secret has been provided to securely interact with OAuth Authorization Server.

Flow:

1. Web Service Client pass its own credentials (credentials it received after registration with OAuth Authorization server) along with other required information to OAuth Authorization sever to get access token.
2. Web Service Client makes call to Resource Server to access protected resource & sends access token along with the request.
3. Resource Servers send access token to OAuth Authorization Server to check the validity of the token.
4. Resource Server gets the descriptive information about the token from OAuth Authorization Server (such as Web Service Client that possesses the access token). Based on this information & additional checks Resource server decides whether to provide access protected content. Resource server sends protected resource's information in response to Web Service Client.

8.5 OpenID Connect Flow



Note:

1. Third party service provider mentioned above could be web application / mobile application / desktop application

Pre-requisite:

1. Third party service provider is registered with OAuth Authorization server. As a part of the registration client_id & client_secret has been provided to securely interact with OAuth Authorization Server.
2. Resource server is also registered with OAuth Authorization Server. As a part of the registration client_id & client_secret has been provided to securely interact with OAuth Authorization Server.
3. User has a valid AccessToken obtained from OAuth Authorization Server using Authorization Code, Resource Owner Credentials, or Implicit Grant Type.
4. User has a valid AccessToken with required openid and profile scope, and optional phone and email scope.

Flow:

1. Resource owner (end user / real user) accesses the third party service provider site. Third party site requires access to the Resource owner's protected resource (Maintained by separate entity called resource server).
2. Service provider sites redirects user to OAuth Authorization Server to get Resource Owner's consent.
3. OAuth Authorization server would make sure Resource Owner is authenticated and provides consent.
4. If Resource Owner provides consent then OAuth Authorization server generates the Authorization Code and sends it to the service provider site's url registered with OAuth Authorization Sever.
5. Third Party Service provider exchanges the Authorization Code with OAuth Authorization Server to receive Access Token & *Refresh Token (provided pre-requisites for generating refresh tokens are met).
6. Third Party Service provider makes call to Resource Server to access Resource Owner's protected resource & sends access token along with the request.
7. Resource Servers send access token to OAuth Authorization Server to check the validity of the token.
8. If required, Resource Servers send access token to OpenID Connect User Info endpoint on OAuth Authorization Server to query consented user attributes.
9. Resource Server gets the descriptive information about the token from OAuth Authorization Server (such as resource owner to which token was issued, third party service provider's identification that possesses the access token). Based on this information & additional checks Resource server decides whether to provide access to resource's owner protected contents to third part service provider. Resource server sends protected resource's information in response to Third party service provider.

9 Development Framework / Libraries

DFT team (dft-mobile@cisco.com) has developed libraries for OAuth Integration. These Libraries provides APIs that can be leveraged by Application Development team for OAuth Integration within their projects.

You can follow below mentioned urls to get more information about DFT Libraries.

M.DFT Services: http://iwe.cisco.com/web/dft/m.dft

OAUTH framework for android:

http://iwe.cisco.com/web/view-post/post/-/posts?postId=213900125

Also if you have any queries you can drop email to dft-mobile@cisco.com mailer alias.

10Reference

- http://documentation.pingidentity.com from Ping Identity
- http://tools.ietf.org/html/rfc6749 - RFC –OAuth 2.0
- http://openid.net/connect/ - OpenID Connect

11Revision History

Revision	Date	Revision Author	Reviewer / Approvers	Description
0.1	12-15-2011	Sean Zhang (xuexzhan)		Initial Document.
0.2	12-17-2011	Sean Zhang (xuexzhan)		Added Section 3. Authentication with Cisco OAuth AS.
0.3	01-10-2012	Sean Zhang (xuexzhan)		Updated Section 2. Creating Cisco OAuth Client IDs and 3.8 Oath Grant Management. Sending to team RFC.
0.4	01-31-2012	Sean Zhang (xuexzhan)		Updated with production endpoint url.
1.0	11-11-2012	Aakash Wasnik (awasnik)	Ranjan Jain (ranjain)	Added OAuth Client API details after PingFed 6.10 upgrade. Restructure Token Creation / Validation / Revocation sections. Added Flow Diagram for grant types.
1.1	01-24-2013	Aakash Wasnik (awasnik)		Added information about DFT libraries
1.2	02-25-2015	Sean Zhang (xuexzhan)	David Ott (dott)	Added OpenID Connect Related Sections