



CA API Management OAuth Toolkit 4.3

Table of Contents

Release Notes	4
Download OTK Installation Files	10
Product Accessibility Features	12
Architecture	14
Installation Workflow	17
Create or Upgrade the OTK Database	17
MySQL Database.....	22
Oracle Database.....	24
Apache Cassandra Database.....	25
Create Database Connections	26
Install the OAuth Solution Kit	28
Dual Gateway Scenario.....	33
Install OTK with API Portal Integration.....	38
Post-Installation Tasks	46
Post-Installation Tasks for the Dual Gateway Scenario.....	50
Update Custom 3.x or 4.0.00 Policies Containing the Decode Json Web Token.....	52
Configure Authentication	54
Token Configuration.....	55
Configure JWT Access Tokens.....	58
Client Authentication.....	65
Create FIP Authentication for Dual Gateways.....	66
Login and Consent Behavior.....	68
Multiple Session Support.....	78
Support Custom Grant Types.....	79
Support Optional Authentication Mechanisms.....	82
Support the SAML Grant Type.....	88
Verify the Installation	90
Run the OAuth 2.0 Test Client.....	92
Verify the OAuth Infrastructure.....	95
Troubleshooting	97
Upgrade the OTK	98
Uninstall the OTK	102
Prepare JSON Message for Export	103
JSON Message Example.....	104
Secure an API Endpoint with OAuth 2.0	109

OAuth Request Scenarios	111
Customizing the OAuth ToolKit	119
Customizing Policies	119
Configure Token Lifetime Properties	120
Client-Specific Customization	121
Configure the Authorization Server	126
Configure PKCE Support	133
Provide Enhanced HTML Form Security	134
Customize Caches	135
Set an Alternative HTTPS Port	141
Registering Clients with the OAuth Manager	145
Manage OAuth Clients with CA API Portal	153
APIs and Assertions	155
OAuth Server API Endpoints	156
OAuth Toolkit APIs	156
OAuth Validation Point (OVP) API	156
CORS Support for OTK APIs	158
OAuth Client Assertions	159
Encapsulated Assertions	162
Error Codes	168
Database Maintenance	198
OpenID Connect Implementation	200
Open ID Connect Implementation Details	201
Generate and Validate an ID Token	205
OpenID Connect Discovery	209
Dynamic Registration	212
Retrieve the JSON Web Key Set (JWKS)	216
Use a Dedicated Private Key for Signing JWT	216
OTK User Role Configuration	220
Tutorials	221
Documentation Legal Notice	222

Release Notes

These Release Notes contain the following details for the CA API Management OAuth Toolkit (OTK):

What's New in OTK Version 4.3

OTK 4.3.1 CR02

The OTK 4.3.1 CR02 release includes OTK 4.3.1 CR01 and additional fixes. The `otk.port` cluster property is now editable and the issue with an unwanted JSON Schema Validation Warning with OTK read-only fragment has been resolved. For more information, see [Resolved Issues](#).

OTK 4.3.1 CR01

The OTK 4.3.1 CR01 release replaces OTK 4.3.1 and includes fixes related to issues with scope issuing vulnerability, `id_token` lookup, and the Client and Client key based custom refresh token. For more information, see [Resolved Issues](#).

OTK 4.3.1

The OTK 4.3.1 contains OTK 4.3 CR01 and additional fixes. For more information, see [Resolved Issues](#).

OTK 4.3 CR01

The OTK 4.3 CR01 release replaces OTK 4.3.00 and contains a critical fix for DE380449. For more information, see [Resolved Issues](#).

OTK 4.3.0

The following sections describe new features, updates to current functionality, and resolutions to known issues. For more information, see [Resolved Issues](#).

Feature/Change	Notes
JWT Access Token Generation and Validation	<p>The OTK now supports issuing and validating two types of access tokens: UUID and JWT. This allows for token validation without the need to call the authorization server.</p> <p>The 256 character length limitation for access tokens has been removed.</p> <p>Note: The JWT access token is not currently supported by the CA Mobile API Gateway.</p> <p>See Token Configuration.</p>

JWT Client Credentials Support	<p>The following functions were added to support client authentication using JWT assertion:</p> <ul style="list-style-type: none"> Two new client authentication methods using JWT are supported: client_secret_jwt and private_key_jwt Client authentication method can be registered for each client through OpenID Connection dynamic registration and through OAuthManager. If not specified, client_secret_basic will be used. OAuth Manager supports either jwks or jwks_uri as valid inputs when a private key JWT is selected as the authentication method. If jwks is specified, JSON is validated. This allows registration and management of clients using private_key_jwt for client authentication. <p>See Dynamic Registration.</p>
Maximum Characters increased for scope, shared secret, and client_id	<p>The scope field now accepts a maximum of 4000 characters. Previously, the maximum was 450 characters.</p> <p>The client_secret field now accepts a maximum of 255 characters. Previously, the maximum was 32 characters.</p> <p>The client_id field now accepts a maximum of 255 characters. Previously, the maximum was 32 characters.</p>
Cassandra Database Performance Improvements	Changes made to improve performance of token queries on Cassandra databases.
Separated Solution Kits for Persistence Layer	<p>The following new solution kits were added to the OAuthSolutionKit.sskar file:</p> <ul style="list-style-type: none"> OTK Assertions – Installs the OTK Assertions server module file that contains custom assertions required by OTK policies Portal Persistence Layer: Cassandra – Enables an OTK with Cassandra to integrate with an API Portal Portal Persistence Layer: MySQL or Oracle – Enables an OTK with MySQL or Oracle to integrate with an API Portal <p>See Install the OAuth Solution Kit.</p>
Cassandra Pagination	<p>Previously, for Cassandra OTK database installations, OAuth Manager support was limited to a single page displaying a maximum number of 100 clients or client tokens.</p> <p>Now, navigation across multiple pages is supported and allows you to view and manage all your clients or tokens. The limit per page remains at 100 entries. For more information, see DE350742 in the Known Issues section.</p>
OAuth 1.0 support has ended	OAuth 1.0 was deprecated in previous releases with limited support. It is now not supported.

Product Compatibility

The following table shows the latest OAuth Toolkit versions and CA product compatibility. All minor versions (CRs) are supported as part of major releases.

CA Mobile API Gateway	CA API Management OAuth Toolkit	CA API Gateway	Mobile SDK for CA Mobile API Gateway	Mobile Developer Console	
4.2	4.3.1, 4.2	10.0, 9.4, 9.3	2.0, 1.9, 1.8, 1.7, 1.6	1.2, 1.1, 1.0***	
4.1*	4.3.1, 4.2	9.4, 9.3	2.0, 1.9, 1.8, 1.7, 1.6	1.2, 1.1, 1.0***	
4.0	4.1, 4.0	9.2	2.0, 1.9, 1.8, 1.7, 1.6, 1.5, 1.4	1.0	

CA Mobile API Gateway	CA API Management OAuth Toolkit	CA API Gateway	Mobile SDK for CA Mobile API Gateway	Mobile Developer Console	
3.3	3.6	9.2, 9.1**	1.9, 1.8, 1.7, 1.6, 1.3	x	
3.2	3.5	9.1	1.9, 1.8, 1.7, 1.6, 1.2	x	

* Requires software compatibility patch. For MAG 4.1 to be compatible with OTK 4.3.1, run the compatibility patch. For more information, see [Download OTK Installation Files](#).

** Cassandra 3.x is not supported in CA API Gateway 9.1.x.

*** Mobile Developer Console (MDC) does not work with OTK 4.3.1

EOS Schedule

EOS Date	Product Versions	
January 6, 2021	OTK 3.x, 4.0, 4.1 MAG 3.x, 4.0, 4.1 SDK 1.x	

OTK 4.3 Database Support

We have certified that OTK works with Community and Standard Editions of MySQL, we recommend you to evaluate Enterprise Edition to get enterprise-grade features, tools, and extensive support for the database from Oracle.

Oracle	MySQL	Cassandra
Oracle 11g	MySQL Server 5.5.x	Apache Cassandra 3.x
Oracle 12c	MySQL Server 5.7.x	

For database scripts, see [Download OTK Installation Files](#).

NOTE

Upgrading to CA API Gateway version 9.3 updates the Gateway database to MySQL Server 5.7.x.

Resolved Issues

OTK 4.3.1 CR02

The following table lists issues that existed in previous releases and are fixed in release OTK 4.3.1 CR02

Reference	Description
DE415409	Resolved an issue where the otk.port cluster property was read-only for new installations. The otk.port cluster property is now editable.
DE398040	Resolved an issue where an unwanted JSON Schema Validation Warning with OTK read-only fragment was generated by the internal Gateway policies and was shown in the application logs.

OTK 4.3.1 CR01

The following table lists issues that existed in previous releases and are fixed in release OTK 4.3.1 CR01

Reference	Description
DE396519	Resolved an issue where under specific conditions, the scope that was granted to an access token was not the scope that the client was registered for.
DE401634	Resolved an issue with custom refresh token. Client and Client key based custom refresh token behavior did not work.
DE394845	Resolved an issue with user authentication. An OTK id_token lookup was performed by default by the OTK User Authentication policy, using the client_id and resource_owner. A call to /oauth/tokenstore/jwt_lookup always failed with a 500 status since there was no JWT in cases where OpenID was not being used.

OTK 4.3.1

The following table lists issues that existed in previous releases and are fixed in release OTK 4.3.1.

Reference	Description
DE365800	Resolved an issue with Single Gateway configuration with Cassandra and server hostname set to localhost. The configuration resulted in failure of getting a new access token using refresh token grant after the previous access token has expired.
DE368517	Resolved an issue where OAuth Client custom field was overwritten when the client was edited from SaaS Portal.
DE373572	Resolved an issue with an Oracle database where users could not query /oauth/clientstore endpoint to retrieve a client using only a client_key.

OTK 4.3 CR01

The following table lists issues that existed in previous releases and are fixed in release OTK 4.3 CR01.

Reference	Description
DE380449	OTK FIP Client Authentication is not failing by default. Refer to the OTK 4.3 CR01 proactive notification for details. The workaround for existing users who have Storage APIs and OAuth Validation Point installed is as follows: <ol style="list-style-type: none"> 1. Open the policy OTK FIP Client Authentication Extension. 2. Enable policy on line 4. It is a disabled assertion 'At least one assertion must evaluate to true'.

OTK 4.3.0

The following table lists issues that existed in previous releases and are fixed in release OTK 4.3.0.

Reference	Description
DE348167	Resolved an issue with incorrect "All assertions must evaluate to true" assertion in /portal/storage API.

Known Issues

The following issues exist in release OTK 4.3.1.

Reference	Description
DE378315	When the OTK database is empty, the OAuth manager fails on login and returns an invalid request error message. Workaround: Populate the database with some test data, either by an API call or a database script and then log in again. For example, the OTK_test_data.sql file from OTK installer will add OAuth test clients. Remove the test clients before going into production.
MST-138 DE242803	OTK returns session status to an unauthorized client. When a user has two client apps registered and one of the apps requests a session status from the server using the other client app's id_token, the server responds with a status 200.
DE241609	Invalid GET method allowed on /oauth/tokenstore/store endpoint.
MST-534	Updating an OAuth client to have the same name and org as an existing client fails as expected, however, a misleading error message occurs.
MST-466	The OAuth Manager accepts invalid JSON content for custom fields if the content starts with valid JSON formatting. The invalid JSON is accepted and saved. No validation error occurs.
MST-50	OTK Token DB Get returns all tokens if token_status is set to empty. Expected behavior is an empty result.
DE350742	When using Cassandra database in OAuthManager, the following combinations are supported. Client_keys filtering and pagination: <ul style="list-style-type: none"> Registered_by, status Client_ident No filters Client_keys filtering with no pagination: <ul style="list-style-type: none"> Registered_by, client_ident Clients filtering and pagination: <ul style="list-style-type: none"> Registered_by No filters Clients filtering with no pagination: <ul style="list-style-type: none"> Name, organization Name Organization Client_ident Client_key Other combinations are not supported.

Additional Resources and Contacts

Click any of the following links for additional information related to the OAuth Toolkit and API Management.

Support

- [CA Support](#)
- [CA API Management Community](#)
- [OAuth Blog on Communities](#)

Related Product Introductions

[API Management Product Pages](#)

Download OTK Installation Files

This topic explains how to download OTK Installation Files.

This page includes:

- Instructions on [how to download](#) the latest CA API Management OAuth Toolkit 4.3.1 CR01 installation file from the support site
- Database files for [creating a new](#) OTK database
- Version-specific database files for [upgrading an existing](#) OTK database to the latest version
- Compatibility patches for [integrating](#) OTK with additional products

This page does not describe how to use these files. For installation instructions, see [Installation Workflow](#).

How to get the CA API Management OAuth Toolkit

The OTK 4.3.1 CR01 release is available at [CA API Gateway Solutions & Patches](#) page.

The entire CA API Management OAuth Toolkit is available as an `OTK_Installers_version-build.zip` file.

For example

The file contains:

- `OAuthSolutionKit-version-build.sskar`
- Database scripts
- Any upgrade Compatibility Patches

To download the OAuth Toolkit:

1. Go to [CA API Gateway Solutions & Patches](#).
2. Click **Sign In** and provide your login credentials.
3. Navigate to the CA API Gateway OAuth Toolkit section.
4. Click the link to download the zip file.

Database Creation and Upgrade Files

Right-click the file links to download the file. Alternatively, you can use the files provided in the download package zip file from the support site.

Create a New Database

Use the following scripts to create a new database. For instructions, follow the [Installation Workflow](#).

Database Creation Scripts and Test Data		
MySQL	Oracle	Cassandra
#unique_9	#unique_11	#unique_13
#unique_10	#unique_12	#unique_14

Upgrade an Existing Database

Use the following scripts to upgrade an existing database. To upgrade an existing database, start with the script corresponding to your current OTK version, then work up the list, executing all scripts until you reach the latest version. For instructions, follow the [Installation Workflow](#).

Individual Database Upgrade Scripts		
MySQL	Oracle	Cassandra
#unique_15	#unique_16	#unique_17
#unique_18	#unique_19	#unique_20
#unique_21	#unique_22	#unique_23
#unique_24	#unique_25	#unique_26 #unique_27 #unique_28
#unique_29	#unique_30	#unique_31 #unique_32
#unique_33	#unique_34	#unique_35
#unique_36	#unique_37	#unique_38
#unique_39	#unique_40	#unique_41
#unique_42	#unique_43	#unique_44
#unique_45	#unique_46	#unique_47
#unique_48	#unique_49	#unique_50
#unique_51	#unique_52	#unique_53
#unique_54	#unique_55	#unique_56
#unique_57	upgrade_otk3.0-otk3.1.1_oracle.sql	#unique_59
#unique_60	#unique_61	
#unique_62	#unique_63	
#unique_64		

Compatibility Patch

The following table provides scripts to use OTK 4.3 with the CA Mobile API Gateway (MAG) version 4.1. Click and download the scripts from the links.

Run the scripts on the MAG database after you have installed the OTK. See [Post-Installation Tasks](#).

OTK 4.2	OTK 4.3
No Patch Required	Run the script that corresponds to your OTK database type. Oracle: #unique_66 MySQL: #unique_67

Product Accessibility Features

CA Technologies is committed to ensuring that all customers, regardless of ability, can successfully use its products and supporting documentation to accomplish vital business tasks.

Product Enhancements

CA Mobile API Gateway offers accessibility enhancements in the following areas:

- Display
- Sound
- Keyboard
- Mouse

Display

To increase visibility on your computer display, you can adjust the following options:

- **Font style, color, and size of items** Defines font color, size, and other visual combinations.
- **Screen resolution** Defines the pixel count to enlarge objects on the screen.
- **Cursor width and blink rate** Defines the cursor width or blink rate, which makes the cursor easier to find or minimize its blinking.
- **Icon size** Defines the size of icons. You can make icons larger for visibility or smaller for increased screen space.
- **High contrast schemes** Defines color combinations. You can select colors that are easier to see.

Sound

Use sound as a visual alternative or to make computer sounds easier to hear or distinguish by adjusting the following options:

- **Volume** Sets the computer sound up or down.
- **Text-to-Speech** Sets the computer's hear command options and text read aloud.
- **Warnings** Defines visual warnings.
- **Notices** Defines the aural or visual cues when accessibility features are turned on or off.
- **Schemes** Associates computer sounds with specific system events.
- **Captions** Displays captions for speech and sounds.

Keyboard

You can make the following keyboard adjustments:

- **Repeat Rate** Defines how quickly a character repeats when a key is struck.
- **Tones** Defines tones when pressing certain keys.
- **Sticky Keys** Defines the modifier key, such as Shift, Ctrl, Alt, or the Windows Logo key, for shortcut key combinations. Sticky keys remain active until another key is pressed.

Mouse

You can use the following options to make your mouse faster and easier to use:

- **Click Speed** Defines how fast to click the mouse button to make a selection.
- **Click Lock** Sets the mouse to highlight or drag without holding down the mouse button.
- **Reverse Action** Sets the reverse function controlled by the left and right mouse keys.
- **Blink Rate** Defines how fast the cursor blinks or if it blinks at all.
- **Pointer Options** Let you complete the following actions:
 - Hide the pointer while typing
 - Show the location of the pointer
 - Set the speed that the pointer moves on the screen
 - Choose the pointer's size and color for increased visibility
 - Move the pointer to a default location in a dialog box

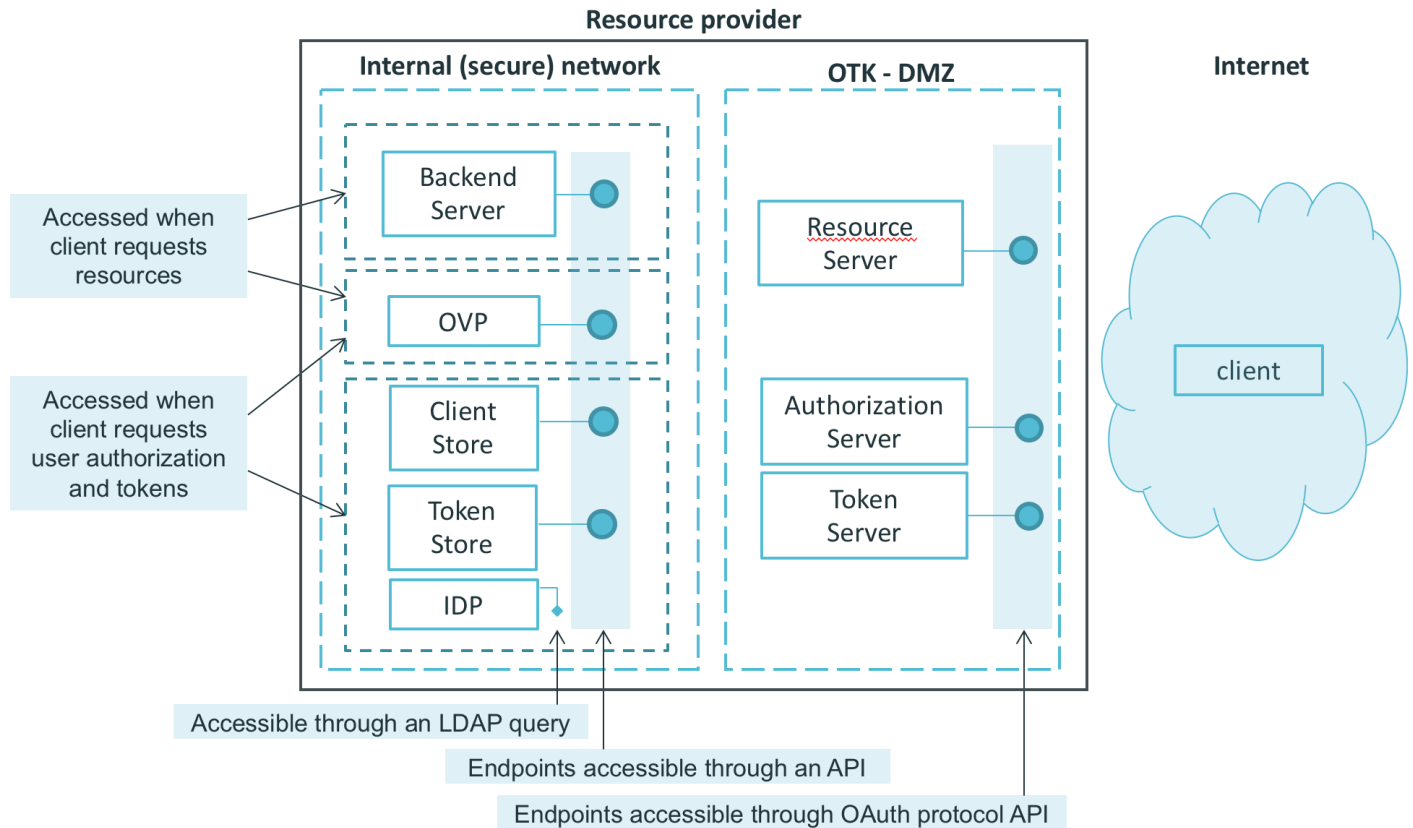
Architecture

The CA API Gateway OAuth Toolkit is separated in the following logically different components.

Component	Notes
OAuth Validation Point (OVP)	An endpoint that validates incoming requests for OAuth 2.0. The endpoint is accessed via a REST API.
DMZ	The CA API Gateway holding the OAuth installation enforcing the OAuth token requirement.
Clientstore	All client_ids are stored here. The clientstore is accessible via a REST API.
Tokenstore	All tokens are stored here. The tokenstore is accessible via a REST API.
Sessionstore	An endpoint that provides caching and session services to the OTK components. This allows OTK components to avoid going to the database in calls to clientstore and tokenstore APIs.
Resource Server	Provides endpoints to access resources. These endpoints require a valid OAuth token.

The following graphic displays the components within their preferred network zones.

OAuth 2.0 OTK layout



Compliance



The CA OAuth Toolkit provides a full featured and standards-compliant OAuth 2.0 solution.

NOTE

OAuth 1.0 is deprecated and no longer supported. Any existing OAuth 1.0 services are removed with an OTK update. No service history is maintained.

OAuth is an authorization standard that allows one service to integrate with another service on behalf of a user. Instead of exposing user credentials, an OAuth access token is issued and accepted for user authentication. The OAuth

authorization framework permits a user to grant an application (consumer) access to a protected resource without exposing the user password credentials.

This implementation conforms to the following specifications:

- **OAuth 2.0:** <http://tools.ietf.org/html/rfc6749>

This implementation may provide incomplete support for the following draft specifications:

- **MAC:** <https://tools.ietf.org/html/draft-ietf-oauth-v2-http-mac-01#section-3.2>
- **Base 64:** <https://tools.ietf.org/html/rfc4648#section-5>



CA API Gateway and CA Mobile API Gateway have been granted certifications for the following OpenID Provider conformance profiles:

- OP Basic
- OP Config
- OP Implicit
- OP Hybrid

These certifications have been registered at OIXnet:

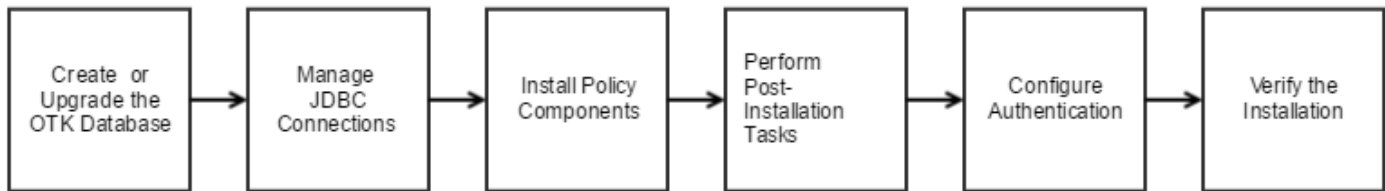
<http://oixnet.org/openid-certifications/>
<http://www.oixnet.org/openid-certifications/ca2/>

NOTE

Specifications can change without notice, possibly causing the OAuth Toolkit to produce incorrect results.

Installation Workflow

Figure 1:



To create the OAuth Tool Kit database and install OTK policy components, follow the OTK installation workflow. The workflow provides instructions for performing both new installations and upgrades.

Perform the following tasks to install and configure the OAuth Toolkit:

- Create or Upgrade the OTK Database
- Create Database Connections
- Install the OAuth Solution Kit
- Post-Installation Tasks
- Configure Authentication
- Verify the Installation
- Troubleshooting

Create or Upgrade the OTK Database

This topic provides links to the create or upgrade procedure for OTK databases, plus model scenarios. The model scenarios present a simplified overview of suggested OTK deployments. Many alternative scenarios are possible.

Before creating or upgrading the OTK database, be aware of the differences between production and development environments, and familiarize yourself with the different types of deployment scenarios (Internal, External, split database, etc.).

How to Create or Upgrade Specific Databases

NOTE

Whether you are creating or upgrading, backup any existing database version before proceeding.

Click any of the following links to access database-specific procedures:

Scripts required for database creation or upgrade are available on [Download OTK Installation Files](#).

See the release notes for compatibility of supported versions.

Production vs Development Environments

Be aware that the scenarios and images presented on this page are simplified They do not show required elements for production environments such as firewalls between the Internal and External zones.

WARNING

In a production environment, do not install the OTK database locally on the Gateway. If the OTK database is installed locally, performance is negatively impacted.

Standard Database Scenarios

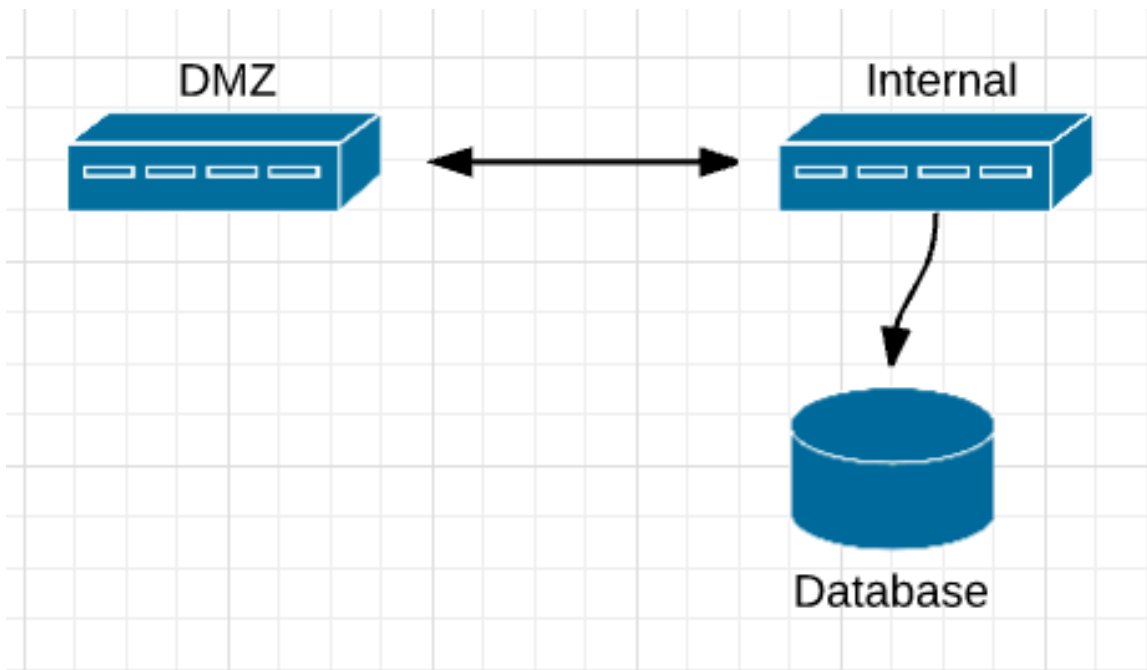
Installation of individual OTK solution kit components for the scenarios is described in [Install the OAuth Solution Kit](#)

Single Gateway, Single Database

This is the default configuration. In a single CA API Gateway scenario, the OTK database is accessed by the Internal Gateway. All OTK components are installed in the same Internal location. By default, OTK policies use a JDBC connection to the database.

Dual Gateway (DMZ - Internal), Single Database

In a dual gateway scenario, there are two instances of the CA API Gateway: the DMZ, and the Internal. OTK components are split across the DMZ and the Internal installations. The OTK database is accessed by the Internal Gateway.



In this scenario, the JDBC connection is between the Internal Gateway and the OTK database.

Modifications Required	
DMZ	Customize OTK policies for OVP configuration, and Storage configuration to point to the Internal Gateway. Import the SSL Certificate from the Internal Gateway

Internal	Customize OTK policies for the authorization server and SAML token authentication to point to the DMZ Gateway. Import the SSL Certificate from the DMZ Gateway.
----------	--

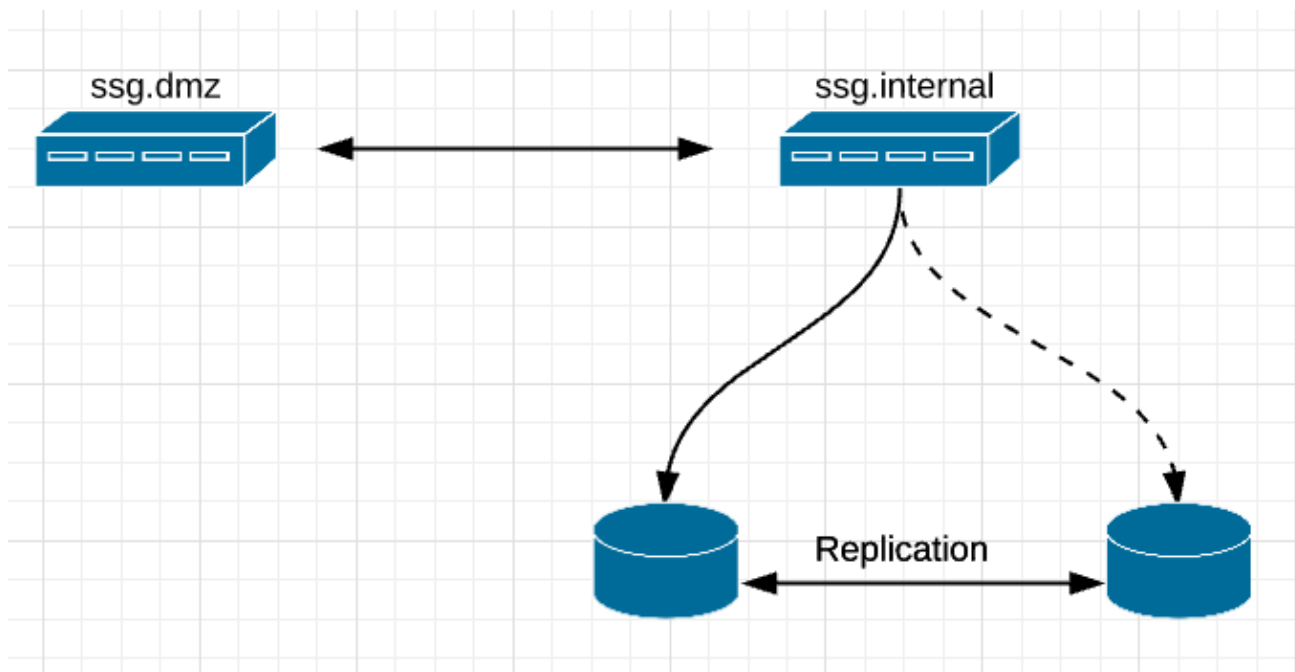
Alternate Database Scenarios

The following diagrams offer alternate database scenarios possibilities. Additional configuration not fully described on this documentation site may be required.

Contact support and the CA Communities site for information.

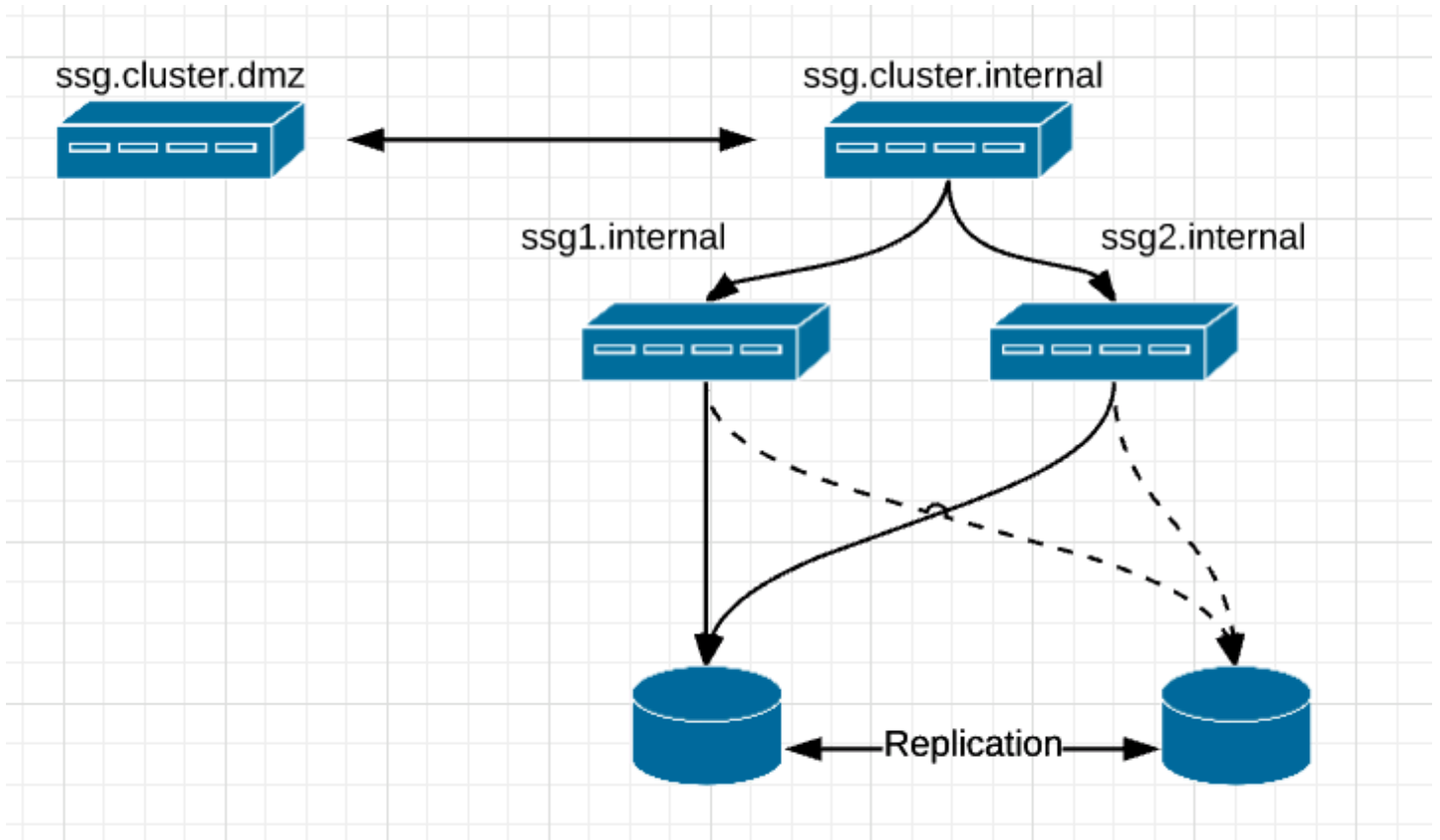
Split Gateway, Database Failover

In this database failover scenario, both databases containing the same data are accessed by the Internal Gateway. You can only have one database connection for each Gateway instance.



Split Gateway, Gateway Cluster, Database Failover

Similar to the previous scenario, this failover scenario includes a Gateway cluster configuration.



Modifications Required	
ssg.cluster.dmz	Customize OTK policies for OVP configuration, and Storage configuration to point to the Internal Gateway. Import the SSL Certificate from the Internal Gateway
ssg.cluster.internal	Customize OTK policies for the authorization server and SAML token authentication to point to the DMZ Gateway. Import the SSL Certificate from the DMZ Gateway.

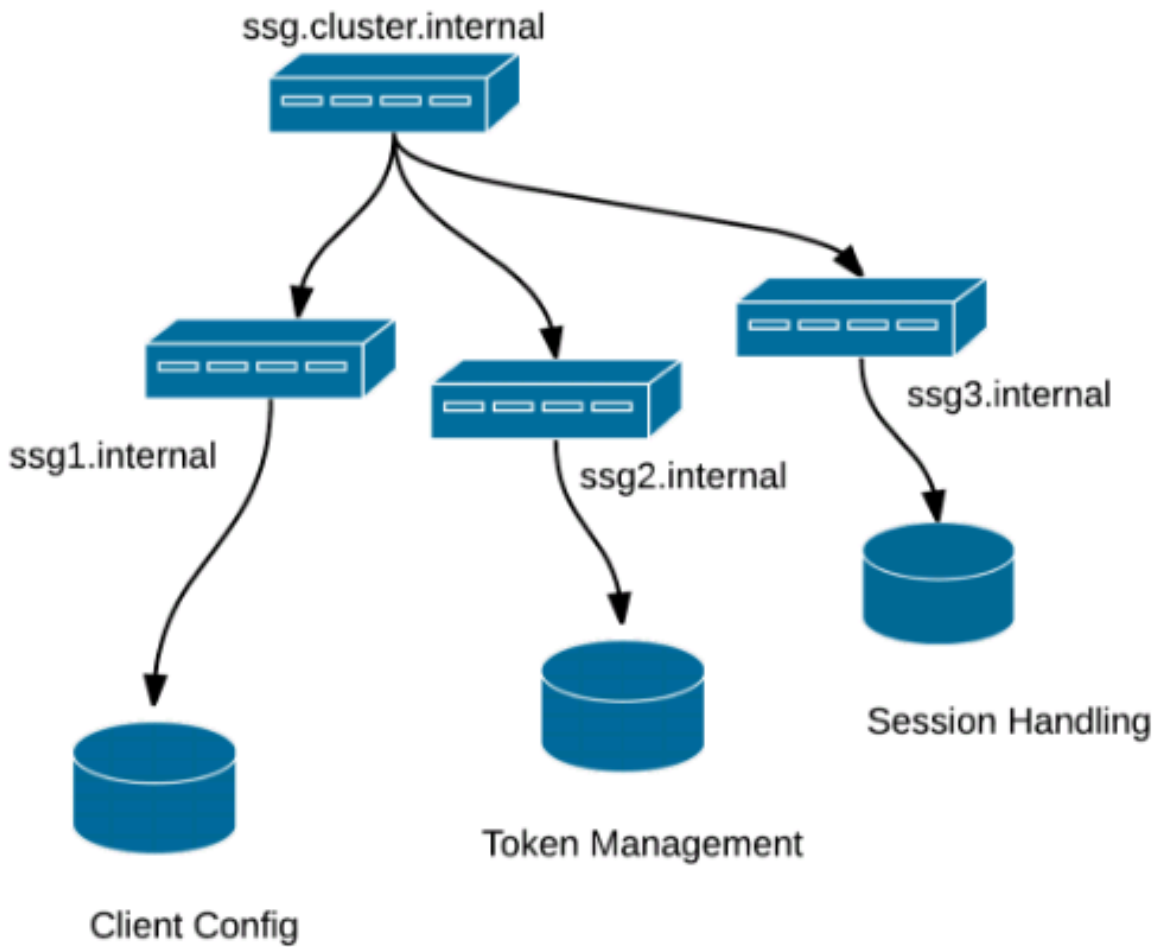
JDBC Connection with failover on ssg.cluster.internal.

```
JDBC URL: jdbc:mysql://ssg1.internal,ssg2.internal:3306/<database-name>?
failOverReadOnly=false&autoReconnect=true&connectTimeout=1
```

Dedicated Databases

Multiple OTK databases can be dedicated to distinct transactions for load balancing.

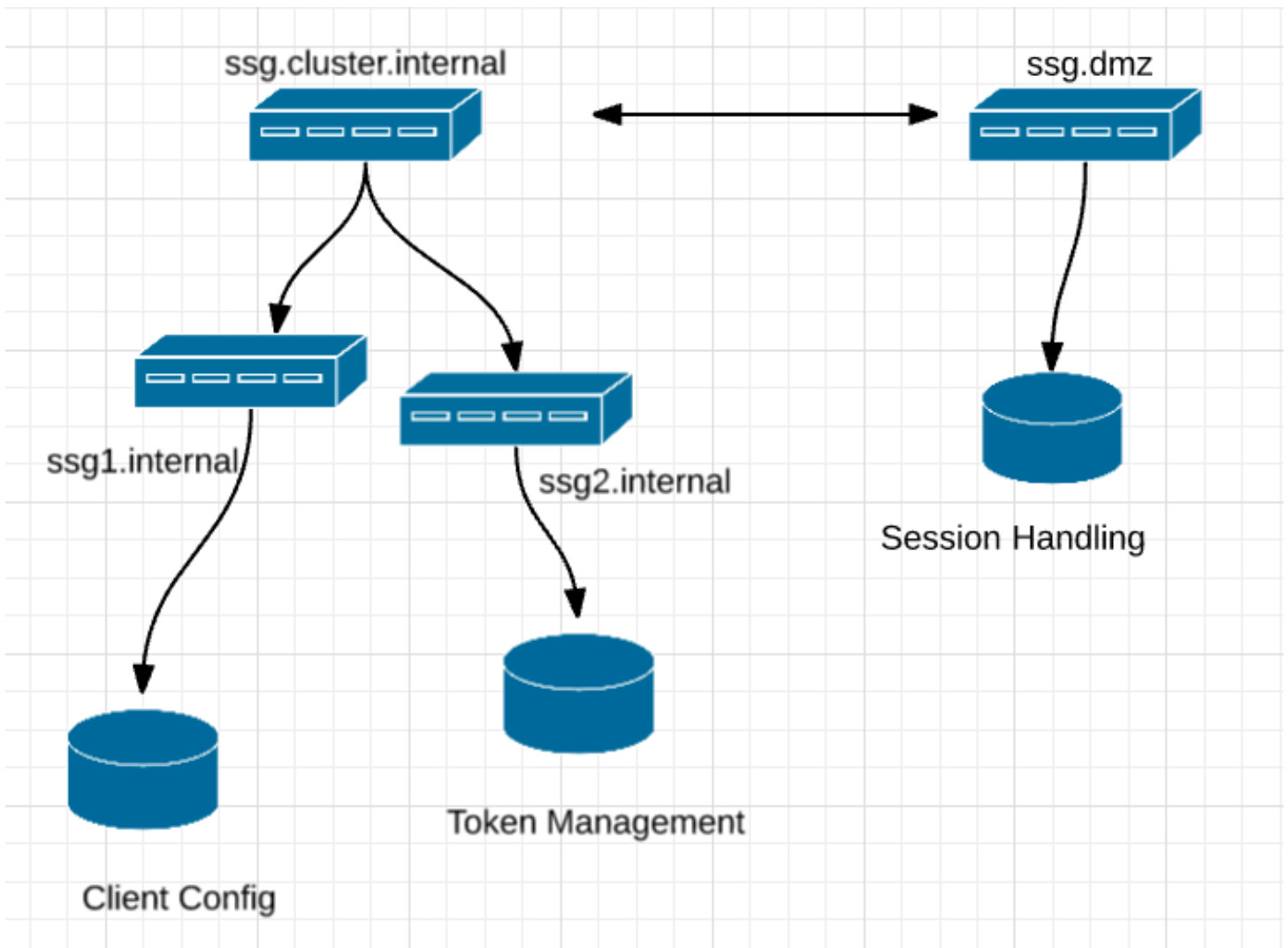
For example, you can dedicate one database OTK for client configurations, one OTK for token management, and another OTK for session handling. To support such a configuration, each OTK instance has its own JDBC or Cassandra connection to the OTK database.



Modifications Required	
<code>ssg1.internal</code>	JDBC or Cassandra connection to the OTK database
<code>ssg2.internal</code>	JDBC or Cassandra connection to the OTK database
<code>ssg3.internal</code>	JDBC or Cassandra connection to the OTK database

Split Gateway, Dedicated Databases

Similarly, the dedicated databases can exist in a split gateway scenario.



MySQL Database

MySQL 5.5.x and 5.7.x Enterprise edition are supported.

Before You Begin

Before creating or upgrading the MySQL database, perform the following tasks:

- Ensure that MySQL is installed on a database host machine.
- Prepare the OTK .sql scripts you require on your local machine.

The .sql script files referenced by these instructions are available for download from this site. See [Download OTK Installation Files](#).

Create the OTK Database

To create the OTK database:

1. Use the mysql program to connect to the server as the MySQL root user.

The following script example creates a database called **otk_db**. Use the syntax that corresponds to your version of MySQL.

MySQL 5.5

```
root@machine_name> mysql
mysql> CREATE DATABASE otk_db;
mysql> GRANT SELECT,UPDATE,DELETE,INSERT ON otk_db.* TO '<db_user>'@'localhost' identified by
  '<db_user_password>';
mysql> flush privileges;
mysql> exit;
```

MySQL 5.7

```
root@machine_name> mysql
mysql> CREATE DATABASE otk_db;
mysql> ALTER USER '<db_user>'@'localhost' identified by '<db_user_password>';
mysql> GRANT SELECT,UPDATE,DELETE,INSERT ON otk_db.* TO '<db_user>'@'localhost';
mysql> flush privileges;
mysql> exit;
```

2. Now run the script to create the schema.

```
root@machine_name> mysql -u root otk_db < otk_db_schema.sql
```

Running `otk_db_schema.sql` creates the schema files and installs test data. If you need to re-install the otk test data, use the file `otk_db_testdata.sql`

You have created the OTK database.

Upgrade an OTK Database

Perform the following tasks to upgrade an existing MySQL OTK database:

NOTE

The following code examples reference a database named **otk_db**.

Back Up Your Database

As a precaution, back up your OTK database before running the upgrade scripts.

Example:

```
[root@ssg]# cd /home/ssgconfig
[root@ssg]# mkdir dbbackup
[root@ssg]# cd dbbackup
[root@ssg]# mysqldump otk_db > otk_backup.sql
```

Delete Existing OAuth Tokens

Optional. Reduce the upgrade script execution time by deleting any existing OAuth tokens. Note that deleting OAuth tokens causes client application users to re-authenticate.

```
mysql> use otk_db;
mysql> DELETE FROM oauth_token;
```

Determine your Current OTK Database Version

To determine the version of your existing OTK database:

1. Connect to your database.

2. As the root MySQL user, run the following command from the mysql shell:

```
mysql> use otk_db;
mysql> select * from otk_version;
```

Run the Upgrade Scripts

Start with the script corresponding to your current OTK version, then work up the list (see [Download OTK Installation Files](#)), executing all scripts until you reach the latest version.

As the database user, run the upgrade scripts from the mysql command line.

For example:

```
mysql> use otk_db;
mysql> source <location>/upgrade_otk4.0.00-otk4.1.00.sql
```

Oracle Database

Before You Begin

Before creating or upgrading the Oracle database, perform the following tasks:

- Ensure that Oracle 12c or 11g is installed on a database host machine
- Prepare the OTK .sql scripts you require on your local machine.

The .sql script files referenced by these instructions are available for download from this site. See [Download OTK Installation Files](#).

Create an OTK Database

To create an OTK database on Oracle:

1. Connect to the Oracle server and start SQL Plus.

For example:

```
cd /u01/app/oracle/product/11.2.0/xe/bin/
source oracle_env.sh
sqlplus
Username: SYSTEM
Password: mypassword
```

2. From the SQL prompt, create a database user.

```
SQL> create user <db_user> identified by <db_user_password>;
SQL> grant connect, resource to <db_user>;
--- Note: CONNECT role enables user to connect to the database
--- Note: RESOURCE role enables user to create certain types of schema objects in that user's own
schema (ie. it grants the create table, but not create view)
SQL> exit
```

3. Run the following command. In this example the sql script is stored in the /temp directory.

```
sqlplus db_user/db_user_password @/temp/otk_db_schema_oracle.sql
```

Running `otk_db_schema_oracle.sql` creates the schema files and installs test data.

If you need to re-install the otk test data, use the file `otk_db_testdata_oracle.sql`

4. **Save.**

You have created the OTK database.

Upgrade an OTK Database

To upgrade an OTK database on Oracle, connect to the Oracle server and start SQL Plus.

Perform the following tasks to upgrade an existing Oracle OTK database:

Back Up Your Database

As a precaution, back up your OTK database before running the upgrade scripts. To create a full backup of an Oracle database, contact your database administrator for support.

Delete Existing OAuth Tokens

Optional. Reduce the upgrade script execution time by deleting any existing OAuth tokens. Note that deleting OAuth tokens causes client application users to re-authenticate.

```
sql> delete from oauth_token;
sql> commit;
```

Determine Your Current OTK Database Version

To determine the version of your existing OTK database:

1. Connect to your database.
2. As the database user, run the following command from SQLPlus:

```
sql> select * from otk_version;
```

Run the Upgrade Scripts

As the database user, run the upgrade scripts from the SQLPlus command line. Download the SQL scripts that you require based on your existing OTK version. SQL scripts are available on the [Download OTK Installation Files](#) page.

For example:

```
sql> start <location>/upgrade_otk4.2.00-otk4.3.00_oracle.sql
```

Apache Cassandra Database

Apache Cassandra™ is an open source non-relational NoSQL database.

The following tasks are for a Cassandra database running on Linux:

- Before you Begin
- Create an OTK Database
- Upgrade a Cassandra OTK Database

Before You Begin

Before creating the Cassandra database, perform the following tasks:

- Ensure that a running Cassandra instance exists.
- Download the script files from the [Download OTK Installation Files](#) page. The scripts must be accessible by the root user.

Create an OTK Database

To create an OTK database on Cassandra:

1. Download the schema and testdata cql files from the [Download OTK Installation Files](#) page. Store the files locally.

2. Log in as the root user to the database node:

```
$ ssh root@yourCassandraDatabase
```

3. Launch the cqlsh shell and create the otk_db keyspace from the prompt:

```
$ cqlsh
cqlsh> CREATE KEYSPACE otk_db WITH replication = {'class' : 'SimpleStrategy',
'replication_factor' : 1};
```

4. From the UNIX command line, run the scripts to create the schema and populate the tables with test data:

```
$ cqlsh -k otk_db -f otk_db_schema_cassandra.cql
$ cqlsh -k otk_db -f otk_db_testdata_cassandra.cql
```

External Files

See the Apache Cassandra documentation for how to run external files.

For example, you can specify an IP Address and port number to start cqlsh on a different node.

Provide user credentials if authentication is required.

```
$ cqlsh 123.123.123.123 9042 -u [username] -p [password] -f otk_db_schema_cassandra.cql
```

Replication Factor

The OTK supports a single-node Cassandra cluster with a replication factor of one.

Upgrade a Cassandra OTK Database

Find the scripts to upgrade a Cassandra OTK database on the [Download OTK Installation Files](#) page.

NOTE

The latest upgrade script has been tested and can copy up to 5 million rows from the old database to the new. If you experience problems running this script, increase PAGETIMEOUT and decrease PAGESIZE values.

Update the Schema

Run multiple scripts sequentially to upgrade your current OTK version to the most recent version.

To upgrade an OTK database on Cassandra:

1. Open an ssh window to a Cassandra node:

```
$ssh root@node.cassandra.myDomain.com
```

2. From the UNIX command line, run the scripts to update the OTK schema. The following example upgrades a Cassandra database from version 4.2.00 to 4.3.00:

```
$ cqlsh -k otk_db -f otk_db_schema_update_4.2.00-4.3.00.cql
```

Create Database Connections

Create and configure the connection to the database. There are two types of data source connections you can create: JDBC connections (for MySQL or Oracle), or Cassandra database connections.


NOTE

We recommend using the default connection names provided: **OAuth** for JDBC connections, **OAuth_Cassandra** for Cassandra connections.

Integration with CA API Portal requires database connections with the default connection names.

Create JDBC Connections

JDBC connections are used with MySQL and Oracle databases.

	<p>To create a JDBC connection:</p> <ol style="list-style-type: none"> 1. From the Policy Manager, navigate to Tasks, Data Sources, Manage JDBC Connections. 2. Click Add or select an existing connection and click Clone. 3. Configure connection properties based on database type. Refer to the sections below. 4. Click Test to verify the JDBC connection works.
---	--

MySQL Database Connection Properties

Create a JDBC connection by configuring properties shown in the following table.

Property	Default Value	Notes
Connection Name	OAuth	Type a name to identify the JDBC Connection. Maximum 128 characters.
Driver Class	com.l7tech.jdbc.mysql.MySQLDriver	Select from the driver classes or provide your own. A support description appears after you select.
JDBC URL	jdbc:mysql://localhost:3306/otk_db	Provide your own URL value.
User Name	otk_user	
Password	password	Replace with a more secure password.

Oracle Database Connection Properties

If you are using an Oracle database, create a connection as shown in the following tables.

Property	Default Value	Notes
Connection Name	OAuth	The name of the JDBC Connection that will be created. Maximum 128 characters.
Driver Class	com.l7tech.jdbc.oracle.OracleDriver	Select from the driver classes or provide your own. A support description appears after you select.
JDBC URL	jdbc:l7tech:oracle:/yourOracleDBServer:1521	
User Name	db_user	

Password	db_user_password	
----------	------------------	--

Additional Properties

Property	Value
Database	<yourDatabaseName>


Support for Multiple Local Databases

Multiple local databases for distinct transactions can be supported. For example, you can dedicate one database for client configurations, one for token management, and another for session handling. To support such a configuration, create one JDBC connection per database and modify the policies to adjust the JDBC assertions to use a non-default JDBC connection. Policies containing JDBC assertions are found in the subfolders of Policy Fragments/persistence/.

Create Cassandra Connections

Apache Cassandra™ does not use JDBC connections.

Create the Cassandra database connection using the following procedure.

	<p>To create the Cassandra connection:</p> <ol style="list-style-type: none"> 1. Navigate to Tasks > Data Sources > Manage Cassandra Connections. 2. Click Add. 3. Configure connection properties as shown in the following table.
--	--

Property	Value	Notes
Connection Name	OAuth_Cassandra	Type a name used to identify the connection.
Contact Points	myCassandra.myCorp.com	Cassandra nodes separated by commas. IP address or DNS
Port	9042	Default value.
Password	dbpassword	
Keyspace	otk_db	Default value. Existing keyspaces can be viewed in cqlsh using "DESCRIBE keyspaces".
Username	root	

Install the OAuth Solution Kit

The OAuth Solution Kit contains the policies, endpoints, and assertions that create the OAuth Toolkit (OTK). From the Policy Manager, install the single OAuth Solution Kit .sskar file. This file contains multiple solution kits that provide specific OAuth functionality.

This page contains the following topics related to installation:

Before you Begin

Perform the following pre-installation tasks:

- Download the OAuth Solution Kit .sskar file
- [Create or Upgrade the OTK Database](#)
- Configure JDBC or Cassandra database connections. See [Create Database Connections](#).

Creating the database connection before installing the solution kits allows you to select the existing connection during the [Resolve Entity Conflicts](#) stage.

Upgrade or Install?

Action	Notes
Upgrade	To upgrade an OTK release to the latest version, follow the upgrade instructions. If upgrading from OTK 4.x, read-only policies are replaced; custom configurations and services are retained. If upgrading from OTK 3.x, install the 4.x version and transfer custom configuration to the #policies.
Install	If you prefer to remove your older installation completely and lose all customizations, perform an Uninstall the OTK , manually delete any remaining folders, then install the new OTK version.
Install and keep previous version	If you prefer to retain a previous version as a reference, install the new OTK version with an instance modifier. We recommend you do not install the OTK with an instance modifier if you intend to integrate with API Portal.

Launch the OAuth Solution Kit Installer

To launch the OAuth solution kit installer:

1. In Policy Manager, go to Tasks, Extensions, and Add-Ons, Manage Solution Kits.
2. If you have an existing OTK version you want to remove, select it and select **Uninstall**. Follow the [uninstall](#) instructions.
Alternatively, you can keep an older OTK version by installing the new version with an instance modifier.
3. Click **Install**.
4. Identify the **Solution Kit File** to use.
Click **File** and locate the signed skar file (.sskar) for the OAuth Solution Kit.
For example: OAuthSolutionKit-4.3.00-1234.sskar.
The path to the solution kit file appears, select **Next**.

The OAuth Solution kit includes multiple solution kits.

Select and Install Solution Kits

The solution kit includes DMZ, Internal, Shared, and Persistence Layer kits.

Choose to install solution kits on the same server, or to split the OTK installation across the DMZ (external) and Internal servers.

NOTE

The recommended multi-server OTK installation is as follows:

- Install the **DMZ** solution kits on the exposed server
- Install the **Internal** solution kits on the protected server
- Install the **Shared** solution kits on both servers
- Install the **Persistence Layer** solution kit that matches your database type on the server that connects to the database. This is usually the **Internal** protected server.

For multi-server installation instructions, see [Dual Gateway Scenario](#).

To select and install specific solution kits on a single server:

1. Select one or more of the available solution kits listed.
Suggested selection is as follows:
 - DMZ, OAuth 2.0, and OpenID Connect endpoints
 - Internal, Server Tools
 - OTK Assertions
 - Persistence Layer: *database type* (where *database type* is the solution kit that matches your currently installed OTK database type)
 - Shared OAuth Resources
2. Assign an optional instance modifier. See [Add an Instance Modifier](#).
3. Click **Next**.

Recommended Base Selection for a Single Server Installation

	Name	Version	... Description
<input checked="" type="checkbox"/>	DMZ, OAuth 2.0 and OpenID Connect endpoints	4.3.1-...	OAuth 2.0 protocol endpoints
<input type="checkbox"/>	Internal, Endpoint to access the client persistence layer	4.3.1-...	Storage, database access to client configuration storage
<input type="checkbox"/>	Internal, Endpoint to access the session persistence layer	4.3.1-...	Storage, database access to session storage
<input type="checkbox"/>	Internal, Endpoint to access the token persistence layer	4.3.1-...	Storage, database access to token storage
<input type="checkbox"/>	Internal, OAuth Validation Point	4.3.1-...	Endpoints that handle protocol related validations
<input type="checkbox"/>	Internal, Portal	4.3.1-...	Portal integration pieces (internal)
<input checked="" type="checkbox"/>	Internal, Server Tools	4.3.1-...	Tools, OAuth Manager and test clients
<input checked="" type="checkbox"/>	OTK Assertions	4.3.1-...	Custom assertions required by OTK files
<input type="checkbox"/>	Persistence Layer: Cassandra	4.3.1-...	Database persistence layer for Cassandra systems
<input checked="" type="checkbox"/>	Persistence Layer: MySQL or Oracle	4.3.1-...	Database persistence layer for MySQL or Oracle systems
<input type="checkbox"/>	Portal Persistence Layer: Cassandra	4.3.1-...	Portal Database persistence layer for Cassandra systems
<input type="checkbox"/>	Portal Persistence Layer: MySQL or Oracle	4.3.1-...	Portal Database persistence layer for MySQL or Oracle systems
<input checked="" type="checkbox"/>	Shared OAuth Resources	4.3.1-...	Resources shared across OTK bundles
<input type="checkbox"/>	Shared Portal Resources	4.3.1-...	Portal integration pieces (common)

Recommended Selection for SaaS CA API Developer Portal Integration

Selection includes the Base selection plus:

- Internal, Portal
- One Portal Persistence Layer selection (depends on database type)
- Shared Portal Resources

	Name	Version	... Description
<input checked="" type="checkbox"/>	DMZ, OAuth 2.0 and OpenID Connect endpoints	4.3.1-...	OAuth 2.0 protocol endpoints
<input type="checkbox"/>	Internal, Endpoint to access the client persistence layer	4.3.1-...	Storage, database access to client configuration storage
<input type="checkbox"/>	Internal, Endpoint to access the session persistence layer	4.3.1-...	Storage, database access to session storage
<input type="checkbox"/>	Internal, Endpoint to access the token persistence layer	4.3.1-...	Storage, database access to token storage
<input type="checkbox"/>	Internal, OAuth Validation Point	4.3.1-...	Endpoints that handle protocol related validations
<input checked="" type="checkbox"/>	Internal, Portal	4.3.1-...	Portal integration pieces (internal)
<input checked="" type="checkbox"/>	Internal, Server Tools	4.3.1-...	Tools, OAuth Manager and test clients
<input checked="" type="checkbox"/>	OTK Assertions	4.3.1-...	Custom assertions required by OTK files
<input type="checkbox"/>	Persistence Layer: Cassandra	4.3.1-...	Database persistence layer for Cassandra systems
<input checked="" type="checkbox"/>	Persistence Layer: MySQL or Oracle	4.3.1-...	Database persistence layer for MySQL or Oracle systems
<input type="checkbox"/>	Portal Persistence Layer: Cassandra	4.3.1-...	Portal Database persistence layer for Cassandra systems
<input checked="" type="checkbox"/>	Portal Persistence Layer: MySQL or Oracle	4.3.1-...	Portal Database persistence layer for MySQL or Oracle systems
<input checked="" type="checkbox"/>	Shared OAuth Resources	4.3.1-...	Resources shared across OTK bundles
<input checked="" type="checkbox"/>	Shared Portal Resources	4.3.1-...	Portal integration pieces (common)

More Installation Options

Choose any of the following installation options:

- Support Alternative Database Scenarios
Alternate database scenarios can require selection of a Persistence Layer solution kit plus configuration of either a JDBC or Cassandra connection.
- Add an Instance Modifier to distinguish this installation from a previous installation of the same version. See [Add an Instance Modifier](#).

Add an Instance Modifier

Do you intend to keep the previous installation as a reference?

If so, select the solution kits and select **Set Instance Modifier**. Type a string value, then select **OK**. The value is added to service resolution URIs, folders, policy names, and other components.

Rules about instance modifiers:

- The instance modifier value must be different for each installation.
- Use the same instance modifier across all the solution kits of a single version.
- If you install OTK with an instance modifier, use the same instance modifier when installing additional products (such as CA Mobile API Gateway).

WARNING

If you intend to integrate with the CA API Developer Portal, do not add an Instance Modifier to your OTK installation. The CA API Developer Portal currently does not support OTK installations with Instance Modifiers.

Resolve Entity Conflicts

The installer tests each solution kit for potential conflicts in the following areas:

- Service routing conflicts
- Policy conflicts
- Certificate conflicts
- Encapsulated Assertion conflicts
- Missing JDBC connections
- Missing assertions

If an error is detected in any of the solution kits:

- The solution kit name is displayed in red.
- The Resolved column for the specific solution kit entity is highlighted and displays "No".
- The **Finish** button is unavailable.

To resolve entity conflicts:

1. Click a solution kit tab highlighted red.
The entities are listed.
2. Select the entity containing the conflict and select **Resolve**. A dialog offers you actions to resolve the conflict.
The Resolved column indicates a resolved conflict.
3. When all conflicts are resolved, select **Finish** to start the installation.
Finish can only be clicked after all conflicts have been resolved.

Resolve the Database Connection Entity Conflict

Selection of the following solution kits requires resolution of the database connection.

- Persistence Layer: MySQL or Oracle
- Persistence Layer: Cassandra
- Shared Portal Resources

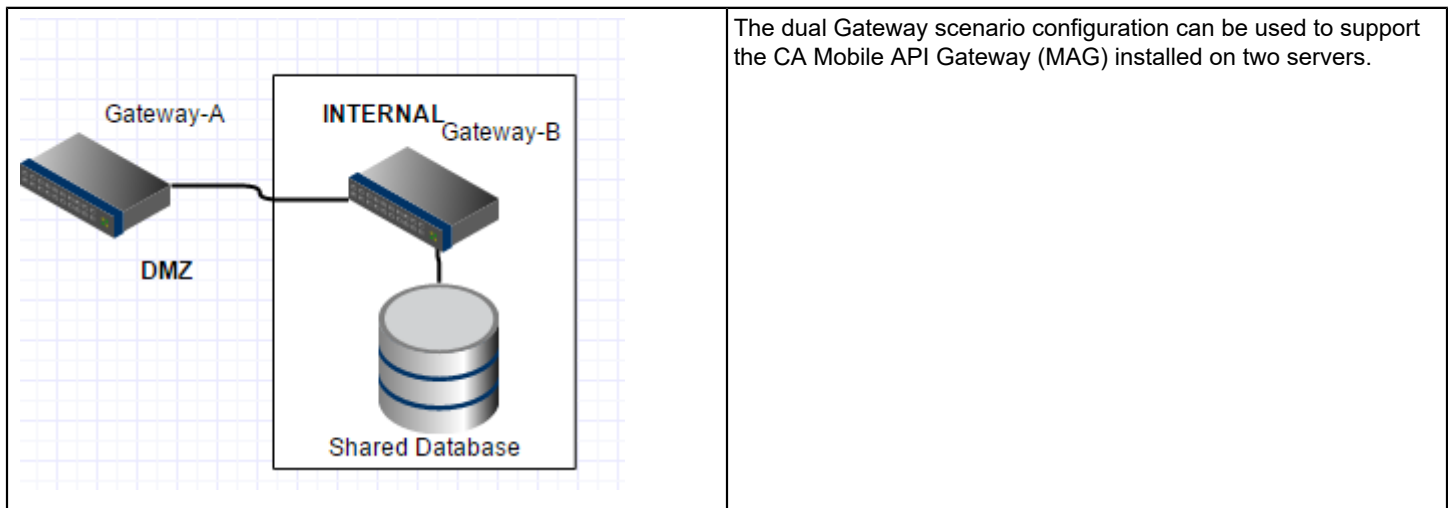
There is an entity conflict to resolve until the OTK database connection is identified. There is an additional database connection entity conflict to resolve if you have selected to integrate with the CA API Management SaaS Portal.

DMZ, OAuth 2.0 and OpenID Connect endpoints		Persistence Layer: Cassandra		Shared OAuth Resources		
Name	Type	Action	Action Taken	Error Type	Resolved	Si
OAuth_Cassandra	CASSANDRA_CONFIGURATION	NewOrExisting	---	TargetNotFo...	No	c2
OTK	FOLDER	NewOrExisting	CreatedNew	---	---	e0

To resolve the database connection entity conflict:

1. Select the entity that requires conflict resolution.
2. Click **Resolve**.
The Resolve Entity Conflict dialog appears.
3. In the Action section, select an existing connection. Otherwise, select **Manage** then **Add** to create a new connection.
4. With the connection selected on the Resolve Entity Conflict dialog, select **OK**.
The entity conflict is resolved. The Resolved column is updated.

Dual Gateway Scenario



The dual Gateway scenario configuration can be used to support the CA Mobile API Gateway (MAG) installed on two servers.

Install Solution Kit Components

The OTK solution kit includes a solution kit components that enable database connection, integration with API Portal, and OAuth functionality.

To make configuration and integration easier:

- Use the default connection names
- Do not assign an instance modifier

Install DMZ Gateway Solution Kit Components

In the recommended dual Gateway scenario, the DMZ Gateway does not connect to the database.

To install the OTK components for the DMZ Gateway:

1. Open the Policy Manager and connect to the DMZ server.
2. Go to Tasks, Extensions and Add-Ons, Manage Solution Kits.
3. Click **Install** and locate the OAuth solution kit sskar file. With the solution kit chosen, click **Next**.
4. Select the following solution kit components:
 - DMZ, OAuth 2.0 and OpenID Connect endpoints
 - Shared OAuth Resources
5. Click **Next**.
6. [Resolve any conflicts](#). If conflicts exist, the solution kit name appears in red and the Finish button is not available.
7. Click **Finish**.

Recommended Solution Kit Component Selection for the DMZ Gateway:

	Name	Version	... Description
<input checked="" type="checkbox"/>	DMZ, OAuth 2.0 and OpenID Connect endpoints	4.3.1-...	OAuth 2.0 protocol endpoints
<input type="checkbox"/>	Internal, Endpoint to access the client persistence layer	4.3.1-...	Storage, database access to client configuration storage
<input type="checkbox"/>	Internal, Endpoint to access the session persistence layer	4.3.1-...	Storage, database access to session storage
<input type="checkbox"/>	Internal, Endpoint to access the token persistence layer	4.3.1-...	Storage, database access to token storage
<input type="checkbox"/>	Internal, OAuth Validation Point	4.3.1-...	Endpoints that handle protocol related validations
<input type="checkbox"/>	Internal, Portal	4.3.1-...	Portal integration pieces (internal)
<input type="checkbox"/>	Internal, Server Tools	4.3.1-...	Tools, OAuth Manager and test clients
<input checked="" type="checkbox"/>	OTK Assertions	4.3.1-...	Custom assertions required by OTK files
<input type="checkbox"/>	Persistence Layer: Cassandra	4.3.1-...	Database persistence layer for Cassandra systems
<input type="checkbox"/>	Persistence Layer: MySQL or Oracle	4.3.1-...	Database persistence layer for MySQL or Oracle systems
<input type="checkbox"/>	Portal Persistence Layer: Cassandra	4.3.1-...	Portal Database persistence layer for Cassandra systems
<input type="checkbox"/>	Portal Persistence Layer: MySQL or Oracle	4.3.1-...	Portal Database persistence layer for MySQL or Oracle systems
<input checked="" type="checkbox"/>	Shared OAuth Resources	4.3.1-...	Resources shared across OTK bundles
<input type="checkbox"/>	Shared Portal Resources	4.3.1-...	Portal integration pieces (common)

More Installation Options

Choose any of the following installation options:

- Integrate with the SaaS CA API Developer Portal. See [Install OTK with API Portal Integration](#).
To integrate with the SaaS CA API Developer Portal, select the **Shared Portal Resources** solution kit component.
- Support Alternative Database Scenarios
Alternate database scenarios can require selection of a Persistence Layer solution kit plus configuration of either a JDBC or Cassandra connection.
- Add an Instance Modifier to distinguish this installation from a previous installation of the same version.
Do not select an instance modifier if you are integrating with the CA API Developer Portal.
See [Add an Instance Modifier](#).

Install Internal Gateway Solution Kit Components

In the recommended dual Gateway scenario, the Internal Gateway connects to the database and provides Storage APIs for OTK on DMZ to access the backend storage.

To install the OTK components for the internal gateway:

1. Open the Policy Manager and connect to the Internal server.
2. Go to Tasks, Extensions and Add-Ons, Manage Solution Kits.
3. Click **Install** and locate the OAuth solution kit sskar file. Click **Next**.
4. Select the following components for your installation:
 - Internal, Endpoint to access the client persistence layer
 - Internal, Endpoint to access the session persistence layer
 - Internal, Endpoint to access the token persistence layer
 - Internal, OAuth Validation Point
 - Internal, Server Tools
 - OTK Assertions
 - Persistence Layer: *database type*
 - Shared Oauth Resources
5. Resolve any conflicts.
Configuration of either a JDBC or Cassandra connection is required.
If conflicts exist, the solution kit name appears in red and the Finish button is not available.
6. Click **Finish**.

Recommended Solution Kit Component Selection for the Internal Gateway

	Name	Version	Description
<input type="checkbox"/>	DMZ, OAuth 2.0 and OpenID Connect endpoints	4.3.1-...	OAuth 2.0 protocol endpoints
<input checked="" type="checkbox"/>	Internal, Endpoint to access the client persistence layer	4.3.1-...	Storage, database access to client configuration storage
<input checked="" type="checkbox"/>	Internal, Endpoint to access the session persistence layer	4.3.1-...	Storage, database access to session storage
<input checked="" type="checkbox"/>	Internal, Endpoint to access the token persistence layer	4.3.1-...	Storage, database access to token storage
<input checked="" type="checkbox"/>	Internal, OAuth Validation Point	4.3.1-...	Endpoints that handle protocol related validations
<input type="checkbox"/>	Internal, Portal	4.3.1-...	Portal integration pieces (internal)
<input checked="" type="checkbox"/>	Internal, Server Tools	4.3.1-...	Tools, OAuth Manager and test clients
<input checked="" type="checkbox"/>	OTK Assertions	4.3.1-...	Custom assertions required by OTK files
<input type="checkbox"/>	Persistence Layer: Cassandra	4.3.1-...	Database persistence layer for Cassandra systems
<input checked="" type="checkbox"/>	Persistence Layer: MySQL or Oracle	4.3.1-...	Database persistence layer for MySQL or Oracle systems
<input type="checkbox"/>	Portal Persistence Layer: Cassandra	4.3.1-...	Portal Database persistence layer for Cassandra systems
<input type="checkbox"/>	Portal Persistence Layer: MySQL or Oracle	4.3.1-...	Portal Database persistence layer for MySQL or Oracle systems
<input checked="" type="checkbox"/>	Shared OAuth Resources	4.3.1-...	Resources shared across OTK bundles
<input type="checkbox"/>	Shared Portal Resources	4.3.1-...	Portal integration pieces (common)

NOTE

Select either **Persistence Layer: Cassandra** or **Persistence Layer: MySQL or Oracle** to match the currently installed database type.

More Installation Options

Choose any of the following installation options:

- Have you already created JDBC or Cassandra database connections? See [Create Database Connections](#). You either select or create the database connection during the [Resolve Entity Conflict](#) stage.
- Integrate with the SaaS CA API Developer PortalTo integrate with the SaaS CA API Developer Portal, select the **Internal, Portal, Shared Portal Resources**, and one of the **Portal Persistence Layer** solution kit components.A JDBC connection for the Shared Portal Resources solution kit is required. Cassandra is not supported.
- Provide an Instance Modifier to distinguish this installation from a previous installation of the same version. See [Add an Instance Modifier](#).

Add an Instance Modifier

Not recommended.

The instance modifier, previously known as the prefix, allows you to install multiple instances of the OTK, each with a unique identifier. The value is added to service resolution URIs, folders, policy names, and other components.

But do you need it?

Before the customization folder was introduced, upgrading the OTK was performed by installing a new instance with an instance modifier, then transferring custom configuration manually from the old to the new version. This method is now obsolete. Any custom configuration is retained in the customization folder when upgrading. As a result, the practical application of the instance modifier is limited. It causes additional configuration for integrating API Portal.

However, if you must add an instance modifier, select the solution kits and click **Set Instance Modifier**. Type a string value, then click **OK**.

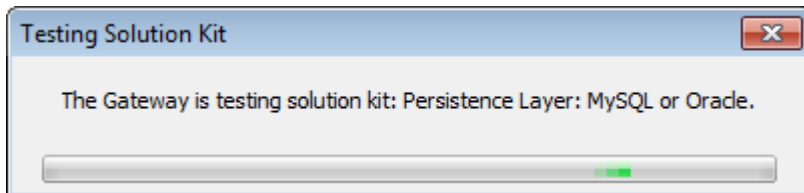
Rules about instance modifiers:

- The instance modifier value must be different for each installation.
- Use the same instance modifier across all the solution kits of a single version.
- If you install OTK with an instance modifier, use the same instance modifier when installing additional products (such as CA Mobile API Gateway).
- Do not use an instance modifier if you want to integrate with API Portal.

Resolve Entity Conflicts

The installer tests each solution kit for potential conflicts in the following areas:

- Service routing conflicts
- Policy conflicts
- Certificate conflicts
- Encapsulated Assertion conflicts
- Missing database connections
- Missing assertions



If an error is detected in any of the solution kits:

- The solution kit name is displayed in red.
- The Resolved column for the specific solution kit entity is highlighted and displays "No".
- The **Finish** button is grayed out and not available.

To resolve entity conflicts:

1. Click a solution kit tab highlighted red.
The entities are listed.
2. Select the entity containing the conflict and click **Resolve**. A dialog box offers you actions to resolve the conflict.
The Resolved column indicates a resolved conflict.
3. When all conflicts are resolved, click **Finish** to start the installation.
Finish can only be clicked after all conflicts have been resolved.

Resolve the Database Connection Entity Conflict

Selection of any the following solution kits requires resolution of the database connection.

- Persistence Layer: MySQL or Oracle
- Persistence Layer: Cassandra
- Portal Persistence Layer: MySQL or Oracle
- Portal Persistence Layer: Cassandra

There will always be an entity conflict to resolve until the OTK database connection is identified. There is an additional database connection entity conflict to resolve if you have selected to integrate with the CA API Management SaaS Portal.

DMZ, OAuth 2.0 and OpenID Connect endpoints		Persistence Layer: Cassandra		Shared OAuth Resources		
Name	Type	Action	Action Taken	Error Type	Resolved	Si
OAuth_Cassandra	CASSANDRA_CONFIGURATION	NewOrExisting	---	TargetNotFo...	No	c2
OTK	FOLDER	NewOrExisting	CreatedNew	---	---	e0

To resolve the database connection entity conflict:

1. Select the entity that requires conflict resolution.
2. Click **Resolve**.
The Resolve Entity Conflict dialog appears.
3. In the Action section select an existing connection. Otherwise, click **Manage** then **Add** to create a new connection.
4. With the connection selected on the Resolve Entity Conflict dialog, click **OK**.
The entity conflict is resolved. The Resolved column is updated.

Post-Installation Tasks for the Dual Gateway Scenario

The following tasks are required after the solution kits are installed:

Restart the Gateway

REQUIRED

After installing the OAuth Solution Kit, restart the CA API Gateway. Failure to restart the Gateway after installation causes errors.

To stop and restart the Gateway using the menu:

1. Access the Gateway main menu.
2. Choose option **2** (Display Gateway configuration menu).
3. Choose option **7** (Manage Gateway status). The current status of the Gateway is displayed.
Press **[Enter]** to continue.
4. Select the option to restart the Gateway. It may take a moment for the Gateway to restart completely.

To stop and restart the Gateway using the command line:

1. Open a privileged shell.
2. Run the following command:
`service ssg restart`

Software Gateway

To stop the Software Gateway:

1. Log in as the gateway or root user.
2. Run the following command:
`/opt/SecureSpan/Gateway/runtime/bin/gateway.sh stop`

To start the Software Gateway:

1. Log in as the gateway or root user.
2. Run the following command:
`# ./runtime/bin/gateway.sh run`

Import the Public Certificate

REQUIRED

The Gateway must trust its own SSL certificate before you can use the test clients. Import the Gateway public certificate into the certificate store of the Gateway.

To import the public certificate:

1. In the Policy Manager, choose **Tasks, Certificates, Keys, and Secrets, Manage Certificates**. The Manage Certificate dialog appears.
2. Click **Add**. The Add Certificate Wizard appears.
3. Select **Retrieve via SSL Connection (HTTPS or LDAPS URL)** and enter *https://<hostname>:8443*.
4. Click **Next**.
The certificate details are displayed.
5. Click **Next**.
On the Specify Certificate Options page, select the following:
 - Outbound SSL Connections
 - Signing Certificates for Outbound SSL Connections
 - Signing Client Certificates
 - Signing SAML Tokens
6. Click **Next**.
Select **Certificate is a Trust Anchor**.
7. Click **Finish** and **Close**.

More Tasks for the Dual Gateway Scenario

REQUIRED

See [Post-Installation Tasks for the Dual Gateway Scenario](#) for detailed procedures to these required tasks:

- Modify Policies on the DMZ Gateway
- Modify Policies on the Internal Gateway
- Configure the id_token Issuer Identifier (Dual Gateway)
- Import SSL Certificates (Both Gateways)
- Configure the FIP Authentication for Dual Gateway

Install OTK with API Portal Integration

API Portal allows owners to control how APIs are published, enables consumers to discover what services are available, and helps operations teams monitor API performance.

Enable API Portal integration with the OTK by installing and configuring API Portal components from the OTK solution kit.

This page contains the following sections:

- Plan of Action
- Select OTK and Portal Solution Kit Components
- Your OTK has an Instance Modifier (Prefix)?
- Post Installation Tasks
- Enroll the Gateway with API Portal

Plan of Action

To Install the OTK with API Portal support:

1. In the Policy Manager, from the OTK solution kit, select both OTK and Portal Solution Kit Components. The selection depends on whether you have a single or dual Gateway scenario.
2. **Do not** assign an instance modifier.
If your OTK already has an instance modifier, you must install another instance of OTK without an instance modifier to support API Portal.
3. Use the default database connection names to avoid additional configuration:
 - "OAuth" for JDBC connections
 - "OAuth_Cassandra" for Cassandra connections.
4. Perform all required post-installation configuration tasks including restarting the Gateway.
5. Enroll with Portal.

The final integration step "Enroll with Portal" assumes that the Portal tenant has been created.

NOTE

API Portal requires an OTK instance **without** an instance modifier (prefix). For new installations, we recommend you install the OTK and the API Portal solution kits without an instance modifier.

If you already have an OTK installed with an instance modifier, you must install a separate instance of the OTK without an instance modifier to work with API Portal. Additional post-installation configuration is required. Future upgrades require upgrading both OTK instances to the new version. See [So your OTK has an Instance Modifier \(Prefix\)?](#)

Select OTK and Portal Solution Kit Components

The OTK sskar file contains all the solution kit components required for OTK and API Portal.

Select the recommended solution kit components corresponding to your single or dual gateway scenario.

For both scenarios, use the default database connection names:

- "OAuth" for JDBC connections
- "OAuth_Cassandra" for Cassandra connections.

Single Gateway Scenario

Selection includes the Base OTK selection plus:

- Internal, Portal
- One Portal Persistence Layer selection (depends on database type)
- Shared Portal Resources

Avoid using instance modifiers.

Recommended solution kit component selection for a single Gateway:

	Name	Version	... Description
<input checked="" type="checkbox"/>	DMZ, OAuth 2.0 and OpenID Connect endpoints	4.3.1-...	OAuth 2.0 protocol endpoints
<input type="checkbox"/>	Internal, Endpoint to access the client persistence layer	4.3.1-...	Storage, database access to client configuration storage
<input type="checkbox"/>	Internal, Endpoint to access the session persistence layer	4.3.1-...	Storage, database access to session storage
<input type="checkbox"/>	Internal, Endpoint to access the token persistence layer	4.3.1-...	Storage, database access to token storage
<input type="checkbox"/>	Internal, OAuth Validation Point	4.3.1-...	Endpoints that handle protocol related validations
<input checked="" type="checkbox"/>	Internal, Portal	4.3.1-...	Portal integration pieces (internal)
<input checked="" type="checkbox"/>	Internal, Server Tools	4.3.1-...	Tools, OAuth Manager and test clients
<input checked="" type="checkbox"/>	OTK Assertions	4.3.1-...	Custom assertions required by OTK files
<input type="checkbox"/>	Persistence Layer: Cassandra	4.3.1-...	Database persistence layer for Cassandra systems
<input checked="" type="checkbox"/>	Persistence Layer: MySQL or Oracle	4.3.1-...	Database persistence layer for MySQL or Oracle systems
<input type="checkbox"/>	Portal Persistence Layer: Cassandra	4.3.1-...	Portal Database persistence layer for Cassandra systems
<input checked="" type="checkbox"/>	Portal Persistence Layer: MySQL or Oracle	4.3.1-...	Portal Database persistence layer for MySQL or Oracle systems
<input checked="" type="checkbox"/>	Shared OAuth Resources	4.3.1-...	Resources shared across OTK bundles
<input checked="" type="checkbox"/>	Shared Portal Resources	4.3.1-...	Portal integration pieces (common)

Dual Gateway Scenario

In the dual gateway scenario example, only the internal gateway accesses the database. Avoid using instance modifiers.

Recommended solution kit component selection for the DMZ Gateway:

	Name	Ve...	Instance Modifier	Description
<input checked="" type="checkbox"/>	DMZ, OAuth 2.0 and OpenID Connect endpoints	4.3....		OAuth 2.0 protocol endpoints
<input type="checkbox"/>	Internal, Endpoint to access the client persistence layer	4.3....		Storage, database access to client configuration storage
<input type="checkbox"/>	Internal, Endpoint to access the session persistence layer	4.3....		Storage, database access to session storage
<input type="checkbox"/>	Internal, Endpoint to access the token persistence layer	4.3....		Storage, database access to token storage
<input type="checkbox"/>	Internal, OAuth Validation Point	4.3....		Endpoints that handle protocol related validations
<input type="checkbox"/>	Internal, Portal	4.3....		Portal integration pieces (internal)
<input type="checkbox"/>	Internal, Server Tools	4.3....		Tools, OAuth Manager and test clients
<input checked="" type="checkbox"/>	OTK Assertions	4.3....		Custom assertions required by OTK files
<input type="checkbox"/>	Persistence Layer: Cassandra	4.3....		Database persistence layer for Cassandra systems
<input type="checkbox"/>	Persistence Layer: MySQL or Oracle	4.3....		Database persistence layer for MySQL or Oracle systems
<input type="checkbox"/>	Portal Persistence Layer: Cassandra	4.3....		Portal Database persistence layer for Cassandra systems
<input type="checkbox"/>	Portal Persistence Layer: MySQL or Oracle	4.3....		Portal Database persistence layer for MySQL or Oracle ...
<input checked="" type="checkbox"/>	Shared OAuth Resources	4.3....		Resources shared across OTK bundles
<input checked="" type="checkbox"/>	Shared Portal Resources	4.3....		Portal integration pieces (common)

Recommended solution kit component selection for the Internal Gateway:

	Name	Ve...	Instance Modifier	Description
<input type="checkbox"/>	DMZ, OAuth 2.0 and OpenID Connect endpoints	4.3...		OAuth 2.0 protocol endpoints
<input checked="" type="checkbox"/>	Internal, Endpoint to access the client persistence layer	4.3...		Storage, database access to client configuration storage
<input checked="" type="checkbox"/>	Internal, Endpoint to access the session persistence layer	4.3...		Storage, database access to session storage
<input checked="" type="checkbox"/>	Internal, Endpoint to access the token persistence layer	4.3...		Storage, database access to token storage
<input type="checkbox"/>	Internal, OAuth Validation Point	4.3...		Endpoints that handle protocol related validations
<input checked="" type="checkbox"/>	Internal, Portal	4.3...		Portal integration pieces (internal)
<input checked="" type="checkbox"/>	Internal, Server Tools	4.3...		Tools, OAuth Manager and test clients
<input checked="" type="checkbox"/>	OTK Assertions	4.3...		Custom assertions required by OTK files
<input type="checkbox"/>	Persistence Layer: Cassandra	4.3...		Database persistence layer for Cassandra systems
<input checked="" type="checkbox"/>	Persistence Layer: MySQL or Oracle	4.3...		Database persistence layer for MySQL or Oracle systems
<input type="checkbox"/>	Portal Persistence Layer: Cassandra	4.3...		Portal Database persistence layer for Cassandra systems
<input checked="" type="checkbox"/>	Portal Persistence Layer: MySQL or Oracle	4.3...		Portal Database persistence layer for MySQL or Oracle systems
<input checked="" type="checkbox"/>	Shared OAuth Resources	4.3...		Resources shared across OTK bundles
<input checked="" type="checkbox"/>	Shared Portal Resources	4.3...		Portal integration pieces (common)

For Dual Gateway scenarios, regardless of API Portal integration, there are additional post-installation tasks. See [Post-Installation Tasks for the Dual Gateway Scenario](#).

So your OTK has an Instance Modifier (Prefix)?

These tasks only apply if your current OTK has an instance modifier. An OTK with an instance modifier is also called as a "prefixed OTK".

For the API Portal to work with a prefixed OTK you must install a separate instance of OTK (without a prefix) then configure your two instances of OTK to work with API Portal.

The tasks include:

- Install a Non-Prefixed OTK Instance
- Remove Duplicate OTK Database Maintenance Tasks
- Create New Prefixed Standard Policy Templates

NOTE

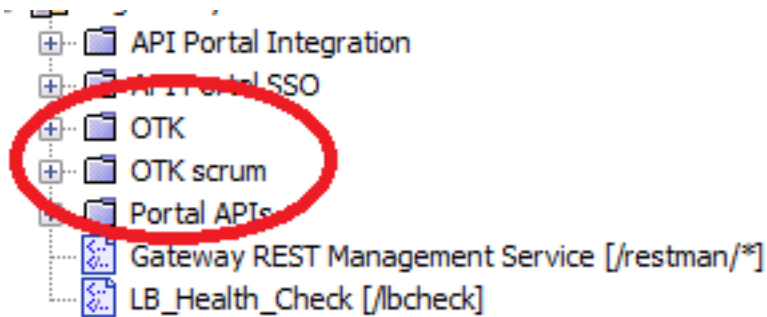
Be aware that the following requirements must be met:

- Both OTK instances must be the same version.
- Both OTK instances must point to the same database

Install a Non-Prefixed OTK Instance

On the Gateway with the prefixed OTK installed, install an non-prefixed instance of the OTK.

For example, the following screenshot shows two OTK instances: one without an instance modifier, and one with the instance modifier "scrum".



Remove Duplicate OTK Database Maintenance Tasks

After installing the OTK instances, there are two sets of **OTK Database Maintenance** tasks: one prefixed, one without a prefix. Because both OTK instances must point to the same database, you only need the non-prefixed set.

1. Go to Tasks > Global Settings > Manage Scheduled Tasks.
2. Select each prefixed OTK Database Maintenance scheduled task, click **Edit**, and click **Remove**.

Manage Scheduled Tasks	
Job Type	Job Name
Recurring	Delete Portal Entities
Recurring	Move Metrics Data Off Box Task
Recurring	OTK Database Maintenance - Client
Recurring	OTK Database Maintenance - id_token
Recurring	OTK Database Maintenance - sessions
Recurring	OTK Database Maintenance - token
Recurring	Portal Bulk Sync Application
Recurring	Portal Check Bundle Version
Recurring	Portal Sync Account Plan
Recurring	Portal Sync API
Recurring	Portal Sync API Plan
Recurring	Portal Sync Application
Recurring	Portal Sync SSO Configuration
Recurring	Portal Tenant Sync Policy Template
Recurring	scrum OTK Database Maintenance - Client
Recurring	scrum OTK Database Maintenance - id_token
Recurring	scrum OTK Database Maintenance - sessions
Recurring	scrum OTK Database Maintenance - token




Create New Prefixed Standard Policy Templates

To create the standard policy templates for the prefixed OTK:

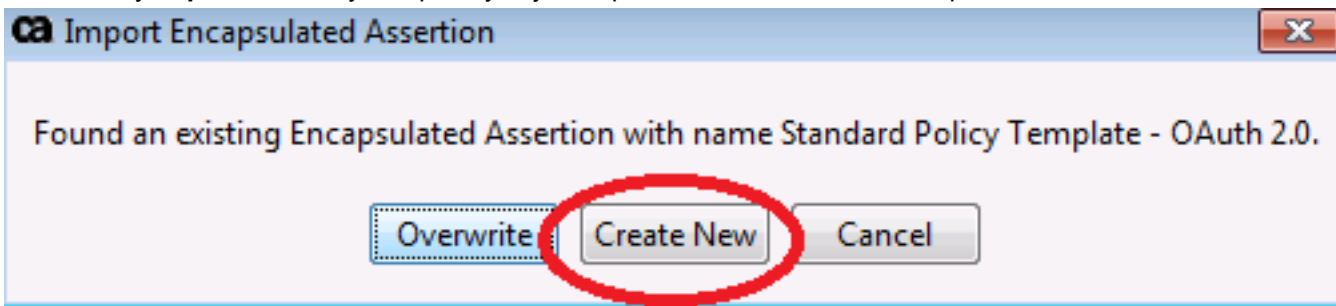
1. Navigate to Tasks > Extensions and Add-Ons > Manage Encapsulated Assertions.
2. Search for "Standard".

A list of Standard Policy Templates encapsulated assertions appears.

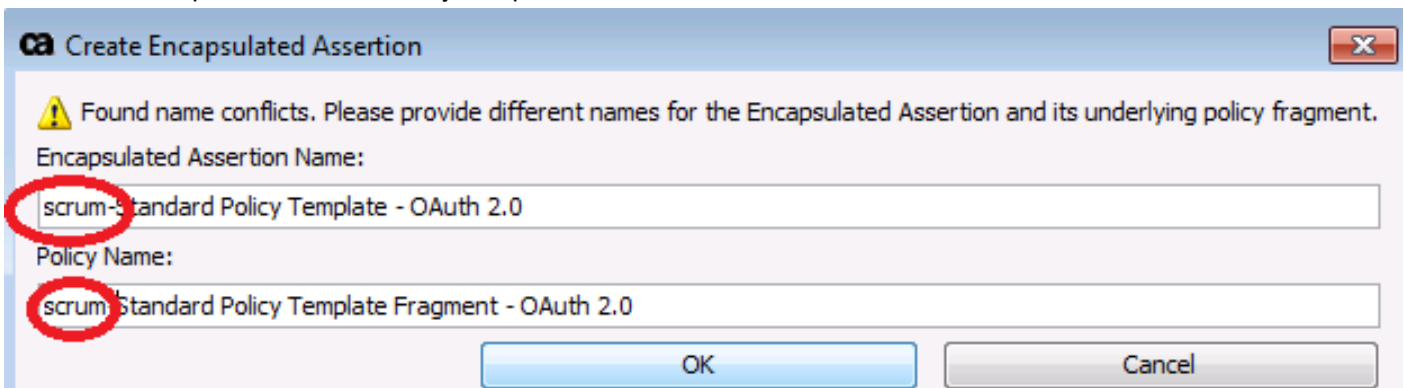
Available Encapsulated Assertions:

	Name
	Standard Policy Template - No Auth
	Standard Policy Template - OAuth 2.0
	Standard Policy Template - API Key

3. Select the template you intend to use for authentication and click **Export**. You can select multiple encapsulated assertions.
4. Immediately **Import** the Policy Template you just exported. When informed of duplication select **Create New**.



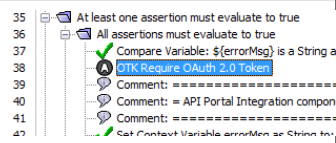
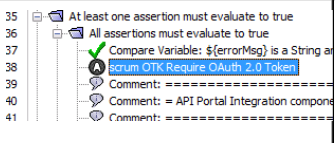
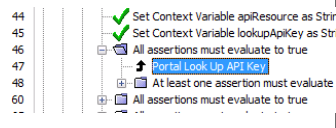
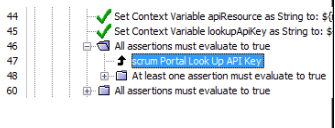
5. Add the prefix to the duplicate references.
Use the format: *prefix*-Standard Policy Template:



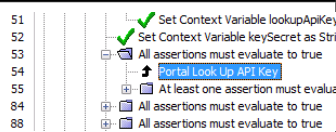
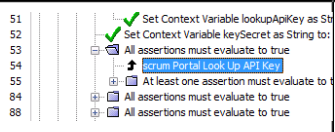
Click **OK**.

A new prefixed Standard Policy Fragment is created.

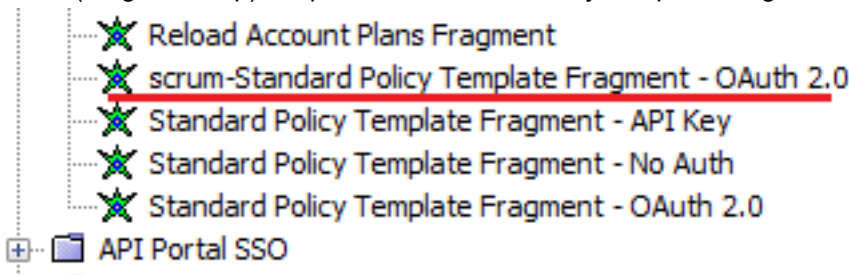
6. In the Policy Manager, locate the newly created prefixed Policy Fragment. Search for the policy fragment by name. The new policy fragment appears in the Policy Fragments folder under the prefixed OTK.
7. Did you create a prefixed **Standard Policy Template Fragment for OAuth 2.0**? If so, open it. Replace the non-prefixed assertions shown in the following table with corresponding prefixed assertions. To make the edits, drag the prefixed assertion from the Assertions panel. Set initial properties as shown in the Notes column, click **OK**, then delete the old assertion.

Where	Original	New	Notes
Line 38			Set Cache validation result to 0.
Line 47			Set the following values: API Key: <code>\${lookupApiKey}</code> Service ID: <code>\${portal.managed.service.apiID}</code>

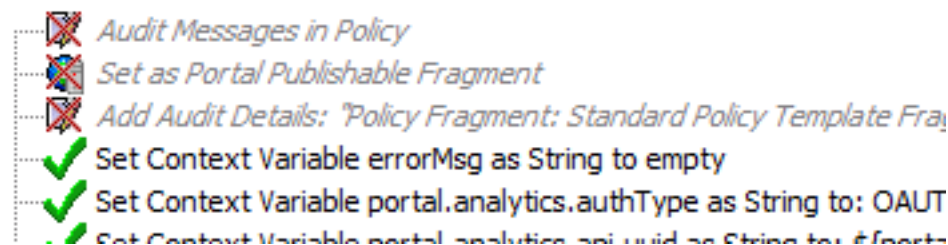
8. Did you create a prefixed **Standard Policy Template Fragment for API Key**? If so, open it. Replace the non-prefixed assertion shown in the table with corresponding prefixed assertion. To make the edit, drag the prefixed assertion from the Assertions panel. Set initial properties as shown in the Notes column, click **OK**, then delete the old assertion.

Where	Original	New	Notes
Line 55			Set the following values: API Key: <code>\${lookupApiKey}</code> Service ID: <code>\${portal.managed.service.apiID}</code>

9. Save the edited prefixed Standard Policy Template Fragments.
10. Move (drag and drop) the prefixed Standard Policy Template Fragments under the API Portal Integration folder.



11. Either remove or disable the duplicated non-prefixed Standard Policy Template Fragment. To disable the non-prefixed fragment: Open the non-prefixed fragment, right-click the "Set as Portal Publishable Fragment" assertion on line 2, select **Disable Assertion**, then **Save**.



Post Installation Tasks

After installing the solution kits, perform the **required** configuration tasks through the Policy Manager.

- [Restart the Gateway](#)
- [Import the Public Certificate](#)

Dual Gateways have additional **required** tasks.

See [Post-Installation Tasks for the Dual Gateway Scenario](#).

Conditionally required tasks may also apply to your installation. Optional tasks are not required and depend on your configuration preferences.

- [Set the Database Type \(Oracle or Cassandra\)](#)
- [Configure the id_token Issuer Identifier \(Dual Gateway\)](#)
- [Configure Support for MAG 4.1](#)
- [Update Custom 3.x or 4.0.00 Policies](#)
- [Create a Dedicated Signing Key for id_token/JWT](#)

Enroll the Gateway with API Portal

An API Portal with a tenant is required before you can enroll. For more detailed instructions, see the "Enroll a CA API Gateway" section of the CA API Portal guide at docops.ca.com/apiportal.

Product version requirements are:

- CA API Gateway 9.2.00 CR05 or higher
- OAuth Toolkit (OTK) version 3.6 or higher

Before You Begin

Ensure that the following requirements are met:

- The tenant record must be created. See [Create the API Portal Tenant](#).
- The Integration tasks for the OTK installation have been performed.
- Ensure that no global policies are configured on the API Gateway.
- Have the API Portal hostname (for example, [apim.mycompany.com](#)) mapped in your DNS server or in the hosts file of your Gateway
- The time on the API Gateway is synchronized with the API Portal. Typically, both entities point to the same NTP server.

We recommend before enrolling the API Gateway, take a snapshot of the API Portal as a backup.

NOTE

if you are not using the default database connection names (**OAuth** and **OAuth_Client** enrollment process fails.

For existing non-default database connection names, clone the db connection and use the default names for the new connections.

DMZ or Internal?

In a dual gateway scenario, we recommend you enroll the Gateway with the DMZ Gateway.

If you enroll with the Internal Gateway, and want to serve APIs from the DMZ Gateway, perform the following tasks:

- Provide developers with a URL to the DMZ Gateway.
- Create service policies that route to the internal gateway. The authorize and token endpoints go to the DMZ Gateway and are routed to the Internal Gateway.

Enroll the Gateway with API Portal

To enrol the Gateway with the Portal server:

1. In a browser, navigate to the new tenant URL that you defined in enroll.json.
2. Log in to the API Portal as the API Portal administrator using the following default credentials:
 - User: admin
 - Password: 7layer
 Change the default password upon login.
3. Select the **Services** icon.
4. Select **Publish > Proxies**.
The API Proxy page displays.
5. Select **Add Proxy**.
6. Enter a name in the **Proxy Name** field.
7. Select Automatic, On Demand, or Scripted deployment type. See API Portal documentation for a description of each.
8. Select **Create**.
The Proxy Enrollment page displays.
9. In **Enrollment URL**, select **Select URL** and copy the value.
10. Using the CA API Gateway Policy Manager, connect to your CA API Gateway.
11. After you are logged in, select **Tasks > Extensions and Add-Ons > Enroll with Portal**.
12. Paste the enrollment URL in the *Enroll with SaaS Portal* window and select **Apply**.
13. Log in to your new tenant Portal, for example, mytenant.mycompany.com , and validate that the external tenant displays.
14. Restart the API Gateway by running


```
service ssg restart
```

 on the API Gateway server.

Post-Installation Tasks

After installing the solution kits, perform the **required** configuration tasks through the Policy Manager.

Conditionally required tasks may also apply to your installation. Optional tasks are not required and depend on your configuration preferences.

NOTE

In the following examples:

- replace *<hostname>* with the hostname of your server. For example, myGateway.com
- replace *<iModifier>* with the instance modifier used when you installed the OTK solution kit. Ignore *<iModifier>* if your Gateway does not have an instance modifier.

Restart the Gateway

REQUIRED

After installing the OAuth Solution Kit, restart the CA API Gateway. Failure to restart the Gateway after installation causes errors.

To stop and restart the Gateway using the menu:

1. Access the Gateway main menu.
2. Choose option **2** (Display Gateway configuration menu).
3. Choose option **7** (Manage Gateway status). The current status of the Gateway is displayed.
Press **[Enter]** to continue.
4. Select the option to restart the Gateway. It may take a moment for the Gateway to restart completely.

To stop and restart the Gateway using the command line:

1. Open a privileged shell.
2. Run the following command:

```
service ssg restart
```

Software Gateway

To stop the Software Gateway:

1. Log in as the gateway or root user.
2. Run the following command:

```
/opt/SecureSpan/Gateway/runtime/bin/gateway.sh stop
```

To start the Software Gateway:

1. Log in as the gateway or root user.
2. Run the following command:

```
# ./runtime/bin/gateway.sh run
```

Import the Public Certificate

REQUIRED

The Gateway must trust its own SSL certificate before you can use the test clients. Import the Gateway public certificate into the certificate store of the Gateway.

To import the public certificate:

1. In the Policy Manager, choose **Tasks, Certificates, Keys, and Secrets, Manage Certificates**. The Manage Certificate dialog appears.
2. Click **Add**. The Add Certificate Wizard appears.
3. Select **Retrieve via SSL Connection (HTTPS or LDAPS URL)** and enter `https://<hostname>:8443`.
4. Click **Next**.
The certificate details are displayed.
5. Click **Next**.
On the Specify Certificate Options page, select the following:
 - Outbound SSL Connections
 - Signing Certificates for Outbound SSL Connections
 - Signing Client Certificates
 - Signing SAML Tokens
6. Click **Next**.
Select **Certificate is a Trust Anchor**.
7. Click **Finish** and **Close**.

Conditionally Required and Optional Tasks

The following tasks are required or optional depending on your configuration and preferences.

Set the Database Type (Oracle or Cassandra)

CONDITIONALLY REQUIRED

This task is only required if you are using an Oracle or Cassandra database. If your database type is MySQL, you do not need to set the database type.

The database type is identified by the `dbsystem` context variable. The default setting for `dbsystem` is "mysql".

To set the database type:

1. In the Policy Manager, open the target policy, **OTK Storage Configuration**. Locate the policy in OTK/Policy Fragments/configuration
2. Copy the Set Context Variable assertion for `dbsystem`.
 - ✔ Set Context Variable `host_oauth_clientstore_server` as String
 - ✔ Set Context Variable `host_oauth_session_server` as String
 - 💬 Comment: NOTE: If the `dbsystem` is Cassandra separate
 - ✔ Set Context Variable `dbsystem` as String to: `mysql`
 - ✔ Set Context Variable `separate_schemas` as String to: `false`
 - 💬 Comment: Max number of allowed Tokens per resource_c
3. Open the hash policy, **#OTK Storage Configuration**. Locate the policy in OTK/Customizations
4. Paste the Set Context Variable assertion for `dbsystem` from the target policy into the hash policy.
5. In **#OTK Storage Configuration**, double-click the assertion. Indicate the database type by typing one of the following values:
 - oracle
 - cassandra
6. Click **OK**, then **Save and Activate** the hash policy.

Hash policies contain custom values that override default values in target configuration policies.

For example, the following code in the **#OTK Storage Configuration** policy sets `dbsystem` to "cassandra". This overrides the default "mysql" setting for `dbsystem` found in the **OTK Storage Configuration** policy.

```

💬 Comment: Target Configuration Policy: "OTK Storage Configuration"
💬 Comment: === Set custom values for Context Variables below ===
💬 Comment: === Add any new Context Variables or extensions below ===
✔ Set Context Variable dbsystem as String to: cassandra

```

Configure the `id_token` Issuer Identifier (Dual Gateway)

CONDITIONALLY REQUIRED

This task is only required in a dual-gateway scenario for the Internal Gateway.

The issuer identifier attribute (`iss`) for `id_token` is set to the URL of the Gateway issuing the `id_token`.

By default, the hostname of the Gateway is used to create the iss URL value: `https://${gateway.cluster.hostname}:8443`

NOTE

In a dual gateway scenario, the iss for id_token must be the URL of the DMZ Gateway. The Internal Gateway iss value must be changed to reference the DMZ gateway hostname in the URL.

To configure the id_token issue identifier:

1. Open the **OTK id_token Configuration** policy. Locate the policy in OTK/Policy Fragments/configuration.
2. Copy the Set Context Variable iss assertion.
3. Open the hash policy, **#OTK id_token Configuration**. Locate the policy in OTK/Customizations
4. Paste the Set Context Variable iss assertion.
5. Double-click the assertion and configure the value to the URL of the Gateway issuing the id_token. In a dual gateway scenario, the iss for id_token must be the URL of the DMZ Gateway.
6. Click **OK**, then **Save and Activate** the hash policy.

Configure Support for MAG 4.1

CONDITIONALLY REQUIRED

This task is required if all the following conditions are met:

- You are running OTK 4.3.x and Mobile API Gateway (MAG) version 4.1.x
- The MAG 4.1 database is either MySQL or Oracle

After installing OTK 4.3 and MAG 4.1, run the MAG compatibility patch against the MAG database (MySQL or Oracle). No action is required if the MAG database is a Cassandra database.

Download the database specific MAG 4.1 Compatibility Patch script from the [Download OTK Installation Files](#) page.

For a MySQL database:

1. Connect to your MAG database.
2. As the database user, run the compatibility script from the mysql command line. In this example the MAG database is mag_db.

```
mysql> use mag_db;
mysql> source <location>/mag_4.1_otk_4.3_compatibility_mysql.sql
```

For an Oracle database:

1. Connect to your MAG database.
2. As the database user, run the script from the SQLPlus command line.

```
sql> start <location>/mag_4.1_otk_4.3_compatibility_oracle.sql
```

Update Custom 3.x or 4.0.00 Policies

CONDITIONALLY REQUIRED

This task is required if all the following conditions are met:

- You upgraded to the latest OTK from OTK 3.x or 4.0.00.
- You created custom policies in OTK 3.x or 4.0.00 that contain the Decode Json Web Token assertion.

See [Update Custom 3.x or 4.0.00 Policies Containing the Decode Json Web Token](#).

Create a Dedicated Signing Key for id_token/JWT

OPTIONAL

By default, the id_token/JWT is signed by the default SSL key. You can use a dedicated private key to perform this task.

See [Use a Dedicated Private Key for id_token Signing](#).

Post-Installation Tasks for the Dual Gateway Scenario

The following tasks apply to a dual Gateway scenario only. They are required after the solution kits are installed:

Modify Policies on the DMZ Gateway

By default, OTK policies support a single gateway scenario and are set to localhost:8443. For the dual gateway scenario, policies on the DMZ must point to the Internal gateway.

Perform the following configuration tasks on the DMZ gateway.

OTK OVP Configuration

To modify the configuration to point to the Internal gateway:

1. On the DMZ gateway, go to OTK/Policy Fragments/configuration
2. Open the **OTK OVP Configuration** policy.
3. Copy the "Set Context Variable host_oauth_ovp_server" assertion.
4. Open the **#OTK OVP Configuration** policy located in the Customizations folder.
5. Paste the assertion and replace "localhost" with the hostname of the **Internal** gateway
6. **Save and Activate**.

OTK Storage Configuration

To modify the configuration to point to the Internal gateway:

1. On the DMZ gateway, go to OTK/Policy Fragments/configuration
2. Open the **OTK Storage Configuration** policy located in the Customizations folder.
3. Copy the "Set Context Variable" assertions for the following:
 - host_oauth_tokenstore_server
 - host_oauth_clientstore_server
 - host_oauth_session_server
4. Open the **#OTK Storage Configuration** policy.
5. Paste the assertions. For each assertion, replace "localhost" with the hostname of the **Internal** gateway.
6. **Save and Activate**.

Modify Policies on the Internal Gateway

By default, OTK policies support a single gateway scenario and are set to localhost. In a dual gateway scenario, certain policies on the Internal Gateway must point to the DMZ gateway. All modification is done inside the **#OTK Client Context Variables** policy located in OTK/Customizations/tools.

To modify the authorization server configuration:

1. On the Internal gateway, search for and open the **OTK Client Context Variables** policy.
2. Copy the "Set Context Variable host_oauth2_auth_server" assertion.
3. Open the **#OTK Client Context Variables** policy.

4. Paste the assertion and replace `${gateway.cluster.hostname}` with the hostname and port of the **DMZ** gateway.
5. **Save and Activate.**

To support SAML Token authentication:

1. On the Internal gateway, go to OTK/Policy Fragments/configuration
2. Open the **#OTK Client Context Variables** policy.
3. Copy the existing Set Context Variable for `host_oauth2_auth_server` assertion. This assertion already has the DMZ hostname as a value.
4. Paste the assertion into the same policy.
5. Modify the Set Context Variable by changing the name to **audience_recipient_restriction**.
6. Remove the port number.
7. Click **OK**.
8. **Save and Activate.**

Configure the id_token Issuer Identifier (Dual Gateway)

Update the `id_token` issuer to match your DMZ gateway URL. See [Configure the id_token Issuer Identifier](#).

NOTE

If you are using a load balancer to expose a port to the external environment, the `id_token` issuer must use the same port.

Import SSL Certificates (Both Gateways)

Each Gateway must import the SSL certificate of the other.

Perform the following tasks:

- Export the SSL Certificate from the DMZ Gateway, and import it into the Internal Gateway
- Export the SSL Certificate from the Internal Gateway, and import it into the DMZ Gateway

To export the SSL certificate:

1. Go to **Tasks > Certificates, Keys and Secrets > Manage Certificates**.
2. Select the certificate with the server host name.
3. Select **Properties**.
4. Click **Export**. Save the certificate using `.pem` format.

To import the SSL certificate:

1. **Tasks > Certificates, Keys and Secrets > Manage Certificates**.
2. Click **Import**. Locate the saved certificate.
3. Click **Load**. The import certificates dialog box appears with the certificate highlighted.
4. In Certificate import options, click **Import as Trust Anchor**.
5. Click **OK**.

Configure the FIP Authentication for Dual Gateway

Ensure that you configure the FIP Authentication otherwise the dual gateway scenario will fail with a client authentication error. See [Create FIP Authentication for Dual Gateways](#).

Update Custom 3.x or 4.0.00 Policies Containing the Decode Json Web Token

REQUIRED*

This

task is required only if you have created custom policies in OTK 3.x or 4.0.00 that contain the Decode Json Web Token assertion. The Decode Json Web Token assertion was modified in OTK 4.0.00 CR01.

Perform the following procedure to ensure that policy processing stops when an invalid JWT signature is detected:

- Search for instances of the Decode Json Web Token assertion
- Add a Compare Expression assertion immediately following the Decode JSON Web Token Assertion and have it test for $\${<prefix>.valid} = \text{true}$. If the JWT signature fails, $\${<prefix>.valid}$ returns 'false' causing the Compare Expression Assertion to fail.

How to Identify Instances of the "Decode JSON Web Token Assertion"

Do the following steps to identify all policies that contain a Decode JSON Web Token Assertion:

1. Access a privileged shell.
2. Run the following command:

```
# mysql ssg -u gateway -p -e "select name from policy where xml like
\">%L7p:DecodeJsonWebToken%\%"
```








3. Make note of the results. They list all the policies and services that contain the assertion.
4. Log in to the Policy Manager either as the administrator or as someone with permissions to modify all policies/services.
5. Open a policy that is identified by the MySQL command above.
6. Press Ctrl-F to invoke the Search bar and then type "Decode Json Web Token" to show all instances of the assertion in the policy.
7. Click a search hit to go to the line in the policy.
8. Add the Compare Expression Assertion in the appropriate place. This is described next.

Where to place the Compare Expression Assertion

Where you place the Compare Expression Assertion depends on the policy logic in use at the branch. These are the main scenarios:

Decode Json Web Token Assertion in Default Policy

In this case, simply place the Compare Expression assertion immediately following the Decode Json Web Token assertion:

```
2 |  Comment: *** Decode JSON Web Token usage examples
3 |  Comment: ***
4 |  Comment: *** Before
5 |  Decode Json Web Token // Assertion by itself
6 |  Comment: *** After adding a comparison assertion
7 |  Decode Json Web Token // Assertion followed by comparison assertion
8 |  Compare Variable: ${output.valid} is equal to true; If Multivalued all values must pass // compare ( ${your-variable-prefix.valid} == true)
```

Decode Json Web Token inside an 'All assertions must evaluate to true' Assertion

Similar to the above, place the Compare Expression assertion within "All assertions...", immediately following the Decode Json Web Token assertion:

```

2  Comment: *** Decode JSON Web Token usage examples
3  Comment: ***
4  Comment: *** Before
5  Decode Json Web Token // Assertion by itself
6  Comment: *** After adding a comparison assertion
7  Decode Json Web Token // Assertion followed by comparison assertion
8  ✓ Compare Variable: ${output.valid} is equal to true; If Multivalued all values must pass // compare ( ${your-variable-prefix.valid} == true)

```

Decode Json Web Token inside an 'At least one assertion must evaluate to true' Assertion

In this instance, you must enclose both the Decode Json Web Token and Compare Expression assertions within a new "All assertions must evaluate to true" assertion.

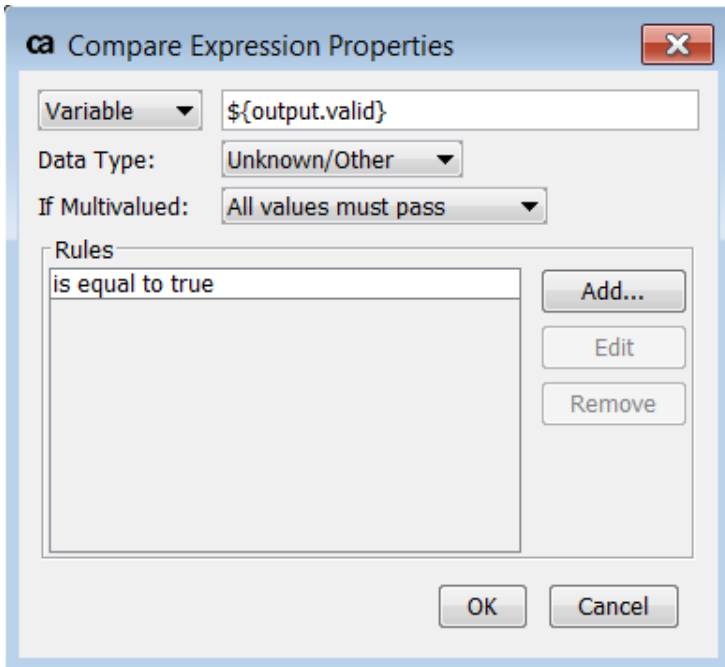
```

2  Comment: *** Decode JSON Web Token usage examples
3  Comment: ***
4  Comment: *** Before
5  ▼ At least one assertion must evaluate to true
6  Comment: Some other assertion
7  Decode Json Web Token // Assertion by itself
8  Comment: Some other assertion
9  Comment: *** After adding a comparison assertion
10 ▼ At least one assertion must evaluate to true
11 Comment: Some other assertion
12 ▼ All assertions must evaluate to true
13 Decode Json Web Token // Assertion followed by comparison assertion
14 ✓ Compare Variable: ${output.valid} is equal to true; If Multivalued all values must pass // compare ( ${your-variable-prefix.valid} == true)
15 Comment: Some other assertion

```

How to Configure the Compare Expression Assertion

Configure the Compare Expression Assertion to test whether the `${<prefix>.valid}` context variable is 'true', where the "<prefix>" is defined in the Decode Json Web Token Assertion. For example, if the prefix is 'output', you define the comparison: `${output.valid} == true`. The properties dialog should look like this:



Configure Authentication

By default, OAuth policies related to client-certificate validation and SAML token-signing validation include sections where authentication is disabled. Attempting to use the endpoints affected by these assertions fails unless manual validation configuration is completed.

Dual Gateway

Policies provided with the CA Mobile API Gateway include endpoints that are used to access storage locations and execute validations. To access these endpoints, the policies require a mutual SSL connection (SSL with client authentication) and verify that the SSL handshake includes a client certificate. Additional manual configuration verifies the client certificate.

Click the workflow image to access configuration instructions.

[Create FIP Authentication for Dual Gateways](#)

Selection of the OAuth validation and storage endpoints during installation is optional. However, if selected, you must [configure client certificate verification](#).

SAML Support

Policies provided with the CA Mobile API Gateway support the SAML 2.0 Bearer Assertion grant type, which uses a SAML token to authenticate users. By default, the OAuth policies validate the SAML token signature. Additional manual configuration verifies that the signature was generated by a trusted party.

Click the workflow image to access configuration instructions.

[Support the SAML Grant Type](#)

If you do not intend to support SAML token, no further action is required.

Token Configuration

The following topics relate to token creation, configuration, and behavior:

Introduction to Token Types

ID Token

ID Token is a token issued as a result of user authentication. For more information about ID tokens, see <http://openid.net/>.

Access Token

Access Token is used by an application to access API on behalf of a user. The two formats of tokens supported in OTK are UUID (default) and JSON Web Token (JWT). For more information about JWT tokens, see [JSON Web Token](#).

Refresh Token

Refresh Token is used to obtain a new access token. It has a longer lifetime than access token. The user does not need to log in every time an access token expires. For customization of the refresh token, see [Configure Refresh Token Behavior](#).

Understanding Access Tokens

In response to a successful client authorization request, the OTK Authorization server generates an access token, which is returned to the client and used to access an API. The access token is consumed by protected resources and is validated for the expiration and status to determine if the request to access to the resource is permitted. The granted scope validation is optional by the configuration in the API.

The OTK supports generation and validation of two types of access token:

UUID Access Tokens

UUID is the default access token format, for backward compatibility. Only the issuing Authorization server can validate the UUID access token. The token can be revoked through the OAuth Manager or a client call to the revocation endpoint. The Authorization server generates UUID formatted access tokens. When an incoming request presents a UUID access token, the database is queried and the token is validated.

Associated policies:

- **OTK Generate OAuth Token** policy – generates a UUID access token.
- **OTK Token Lifetime Configuration** policy – contains the default setting for the `oauth2_access_token_lifetime` context variable.
- **#OTK Token Lifetime Configuration** policy – provides customization of the settings in the OTK Token Lifetime Configuration policy.

JWT Access Tokens

NOTE

The JWT access token is now supported for applications built on CA Mobile API Gateway.

The characteristics of the JWT access token are:

- The JWT is signed with a private key and can be validated without calling the authorization server
- Includes more information than a UUID token
- Claims in JWT payload are visible to clients (JSON format)
- Resource server and Clients can verify the token using the RS256 signing algorithm
- JTI in the JWT can be used as a UUID token

For more information on how to enable JWT access token, how the validation is performed, and how to disable calls to authorization server for validation, go to [Configure JWT Access Tokens](#).

Example of a JWT access token:

The header of the JWT contains the kid claim value which references a public key that corresponds to the public JWK (JSON Web Key).

```
{
  "typ": "jwt",
  "alg": "RSA256",
  "kid": "key_id_of_used_private_key"
}
```

The payload of the JWT contains the claims including the scopes and the JTI (JWT ID).

```
{
  "iss": "https://example.com",
  "iat": 1519860220,
  "aud": "63e8c4b0-dbdf-4b99-8551-2f2b0bcd80ab",
  "exp": 1519863820,
  "jti": "d5ff6a3c-5744-488b-8882-54e5532db5f9",
  "token_details": {
    "scope": "openid email profile openid_client_registration",
    "expires_in": 3600,
    "token_type": "Bearer"
  }
}
```

Configure Refresh Token Behavior

You can customize the reuse and expiration behavior of the refresh token.

The default behavior is as follows:

1. A refresh token is for one-time use only.
2. When a refresh token is used, a new refresh token is issued with a new expiration value.

To customize the default behavior for all refresh tokens:

1. In the Policy Manager, open the **OTK OVP Configuration** policy. Locate the policy in OTK/Policy Fragments/configuration/
2. Copy the following assertions:
 - a. Set Context Variable reuse_refresh_token
 - b. Set Context Variable reuse_refresh_expiration
3. Open the **#OTK OVP Configuration** policy. Locate the policy in OTK/Customizations/
4. Paste the assertions.
5. Double-click each assertion and set the Expression field to **true** or **false**.

6. Click **OK**.
7. **Save and Activate** the policy.

Context Variable	Notes
reuse_refresh_token	<p>Either true or false.</p> <p>If false, the refresh token is for one-time use only within the configured expiration time. After the refresh token is used, the token is deleted. A new refresh token is issued.</p> <p>If true, the same refresh token can be reused multiple times until the configured expiration time.</p>
reuse_refresh_expiration	<p>Either true or false.</p> <p>Determines whether a new or the original expiration time is used for the refresh token.</p> <p>If false, when a refresh token is used, a new expiration time is issued.</p> <p>If true, the original expiration time is maintained for any newly issued or reused refresh token.</p>

Set the Maximum Token Count

Set the maximum number of tokens that are allowed per resource owner and client. When the maximum number is exceeded, either deny the request and return an error, or cycle the tokens by adding the new token and removing the oldest.

For example,

1. With the **max_oauth_token_count** set to 5, all clients can access up to five instances of the same app without logging out of the first instance.
2. When a client attempts to log in to more than five instances, the **max_oauth_token_behaviour** setting is applied. If set to cycle, the client is logged out of the first instance and logged into a new instance. If set to error, the client is not logged into the new instance and an error is returned.

To set the maximum number of allowed tokens per resource owner and client:

1. Open the OTK Storage Configuration policy. Find the policy in OTK/Policy Fragments/configuration.
2. Copy the following assertions:
 - a. Set Context Variable max_oauth_token_count
 - b. Set Context Variable max_oauth_token_behaviour
3. Open the #OTK Storage Configuration policy. Find the policy in OTK/Customizations.
4. Paste the assertions.
5. Double-click the assertions and modify the default values.
6. Click **OK**.
7. **Save and Activate**.

Context Variable	Notes
max_oauth_token_count	The number of OAuth tokens that are allowed per resource owner and client application.
max_oauth_token_behaviour	<p>Either cycle or error.</p> <p>If cycle, the oldest token is removed and the new token is issued.</p> <p>If error, the new token is not issued. An error is returned.</p>

Implement Client-Specific Configuration

You can configure token behavior for a specific client.

For example, to customize the reuse refresh token behavior for a specific client, open OAuth Manager, select the client, and add context variables to the custom field.

```
{
  "reuse_refresh_token":"true"
  "reuse_refresh_expiration":"true"
}
```

Further policy changes are required to capture the custom values. See [Client-Specific Customization](#).

Configure JWT Access Tokens

This page explains how to generate and configure OAuth Access Tokens as JSON Web Tokens (JWTs).

Associated policies:

- **OTK Generate JWT OAuth Token** – Generates a JWT access token
- **#OTK Generate JWT OAuth Token** – Allows you to customize the JWT access token. Includes sample code that inserts a preferred user value into the JWT payload if the scope includes OpenID.

Generate a JWT Access Token

By default, the Authorization server generates UUID formatted OAuth Access tokens. The following instructions show how to enable the Authorization server to issue an OAuth access token in JWT format. Custom configuration is performed within the policy by setting context variables and enabling or disabling assertions.

To enable the Authorization server to generate JWT Access tokens:

1. In Policy Manager, open the **#OTK Generate JWT OAuth Token** policy.
By default, the assertions in this policy are disabled.
2. Enable all assertions in the policy. You can select multiple assertions, right-click, and select Enable Assertion.
3. Disable the *All assertions must evaluate to true* folder. This folder provides additional configuration for OpenID Connect. See **Add an Additional Claim to the JWT Payload** for more details.

----- Comment: Policy Fragment: #OTK Generate JWT OAuth Token

----- Comment: Target policy: OTK Generate JWT OAuth Token

----- Comment: Enable the below assertions to generate JWT based access_token

----- Comment: Variable 'custom' includes session information and preferred user

----- ✓ Set Context Variable exp as Date/Time to empty

----- ✓ Set Context Variable at_jwt_exp as String to: `${exp.seconds}`

----- ✓ Set Context Variable custom_json as Message to: `${custom_data}`

----- Comment: Below 'All Assertion' includes preferred user into the JWT payload if scope includes 'openid'

-----] At least one assertion must evaluate to true

----- Comment: Disable the 'All Assertion' below if preferred user need not be included in payload

----- **✗ All assertions must evaluate to true**

----- ✓ Set Context Variable payload as String to: `{ "iss": "${iss}", "iat":${gateway.time.seconds}, ...`

----- Encode Json Web Token: sign payload

----- ✓ Set Context Variable at_jwt as String to: `${at_jwt.compact}`

4. **Save and Activate** the policy.

The following example shows the default JWT access token claims:

Context Variable	Notes
exp	The date and time the access token expires.
payload	<pre>{ "iss": "\${iss}", "iat": \${gateway.time.seconds}, "au": "\${aud}", "exp": \${at_jwt_exp}, "jti": "\${jti}", "token_details": { "scope": "\${scope}", "expires_in": \${lifetime}, "token_type": "Bearer" } }</pre>

Add Custom Claims to the JWT Payload

Custom claims are used to provide additional information to the protected API for validation or access control. Claims can be added to the JWT as long as the payload maintains a valid JSON format.

NOTE

If you add claims to the JWT payload, do not include any sensitive information that clients should not see, such as password.

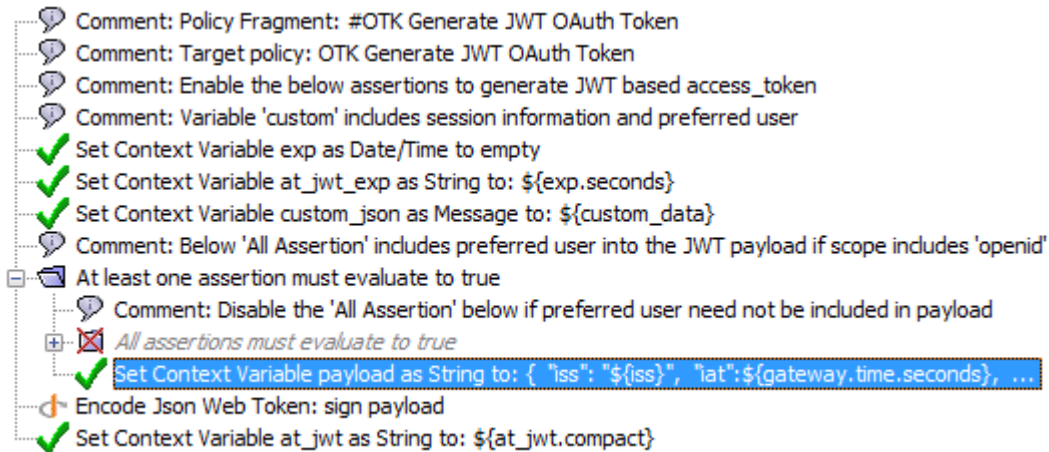
The procedure to add additional claims is as follows:

- Pass in the value
- Extract the value
- Set a value to the payload context variable

Add a Custom Claim

To add a custom claim to the default JWT Payload:

1. Prepare the #OTK Generate JWT OAuth Token policy as described above. Leave the *All assertions must evaluate to true* folder disabled.
2. Double-click the "Set Context Variable payload..." assertion.



The payload properties appear with the default claims expressed in JSON format.

3. Edit the Expression field, adding your custom claims. Click **OK**.

Variable Name:

OK (Overwrite)

Data Type:

Content-Type:

Expression:

```
{
  "iss": "${iss}",
  "iat": ${gateway.time.seconds},
  "aud": "${aud}",
  "exp": ${at_jwt_exp},
  "jti": "${jti}",
  "token_details": {
    "scope": "${scope}",
    "expires_in": ${lifetime},
    "token_type": "Bearer"
    "custom_claim": "${value}"
  }
}
```

4. **Save and Activate** the policy.


Add an OpenID Connect Custom Claim

The disabled folder in the #OTK Generate JWT OAuth Token policy contains assertions that check that OpenID Connect is in scope (openid), and add the preferred_username claim. The preferred_username claim allows access to the userinfo endpoint: openid\connect\v1\userinfo.

Use the example to add any OpenID Connect dependent claim.

To set the preferred_username claim:

1. Prepare the #OTK Generate JWT OAuth Token policy as described above.
2. Enable the *All assertions must evaluate to true* folder. The preferred_username claim is pre-configured.
3. **Save and Activate**.

Variable Name:	payload
	 OK
Data Type:	String
Content-Type:	text/xml; charset=utf-8
Expression:	<pre>{ "iss": "\${iss}", "iat": \${gateway.time.seconds}, "aud": "\${aud}", "exp": \${at_jwt_exp}, "jti": "\${jti}", "token_details": { "scope": "\${scope}", "expires_in": \${lifetime}, "token_type": "Bearer", "preferred_username": "\${owner.result}" } }</pre>

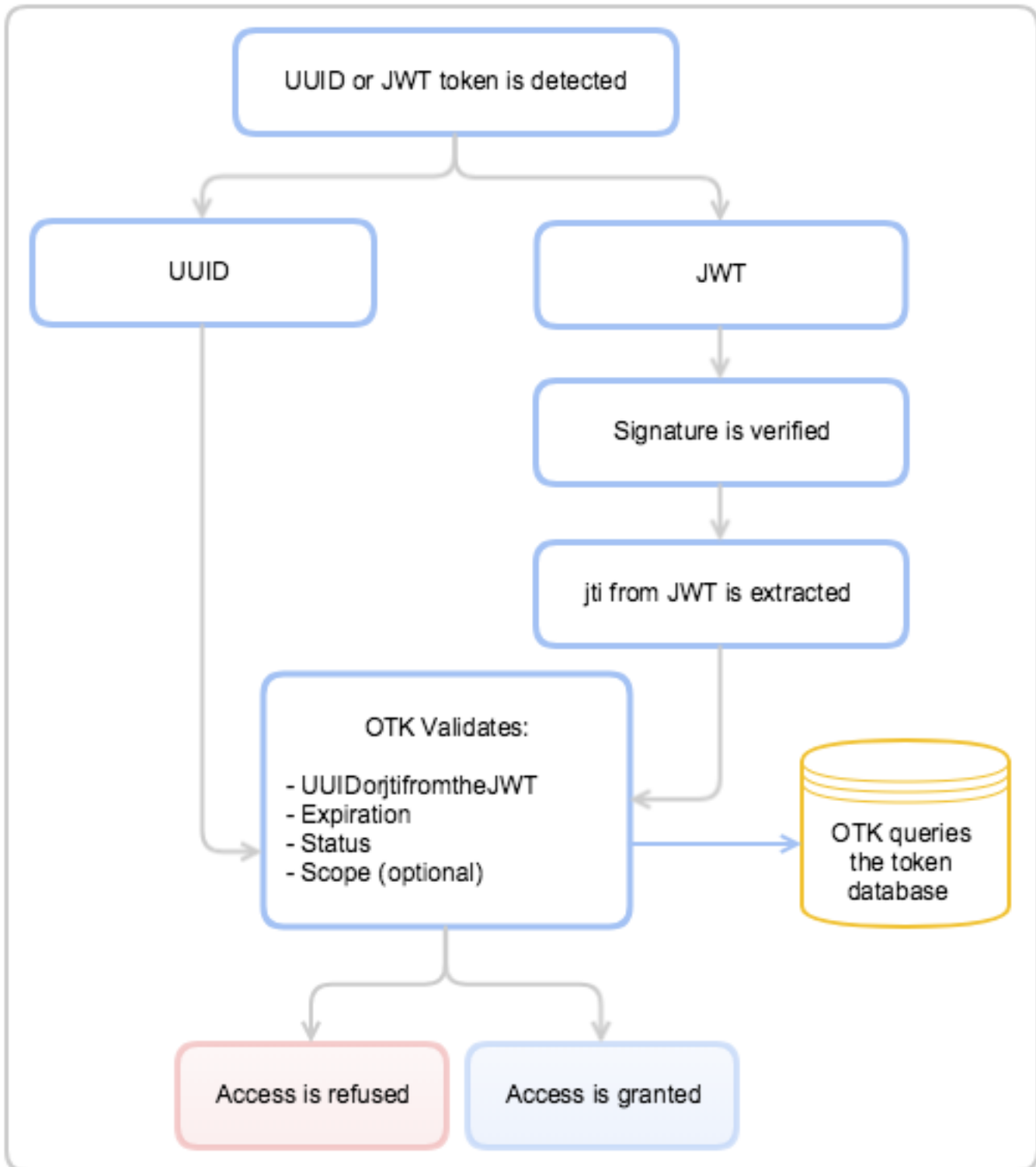
Validate a JWT Access Token

You can validate a JWT Access Token with or without querying the Authorization Server.

Validate With the Authorization Server Database

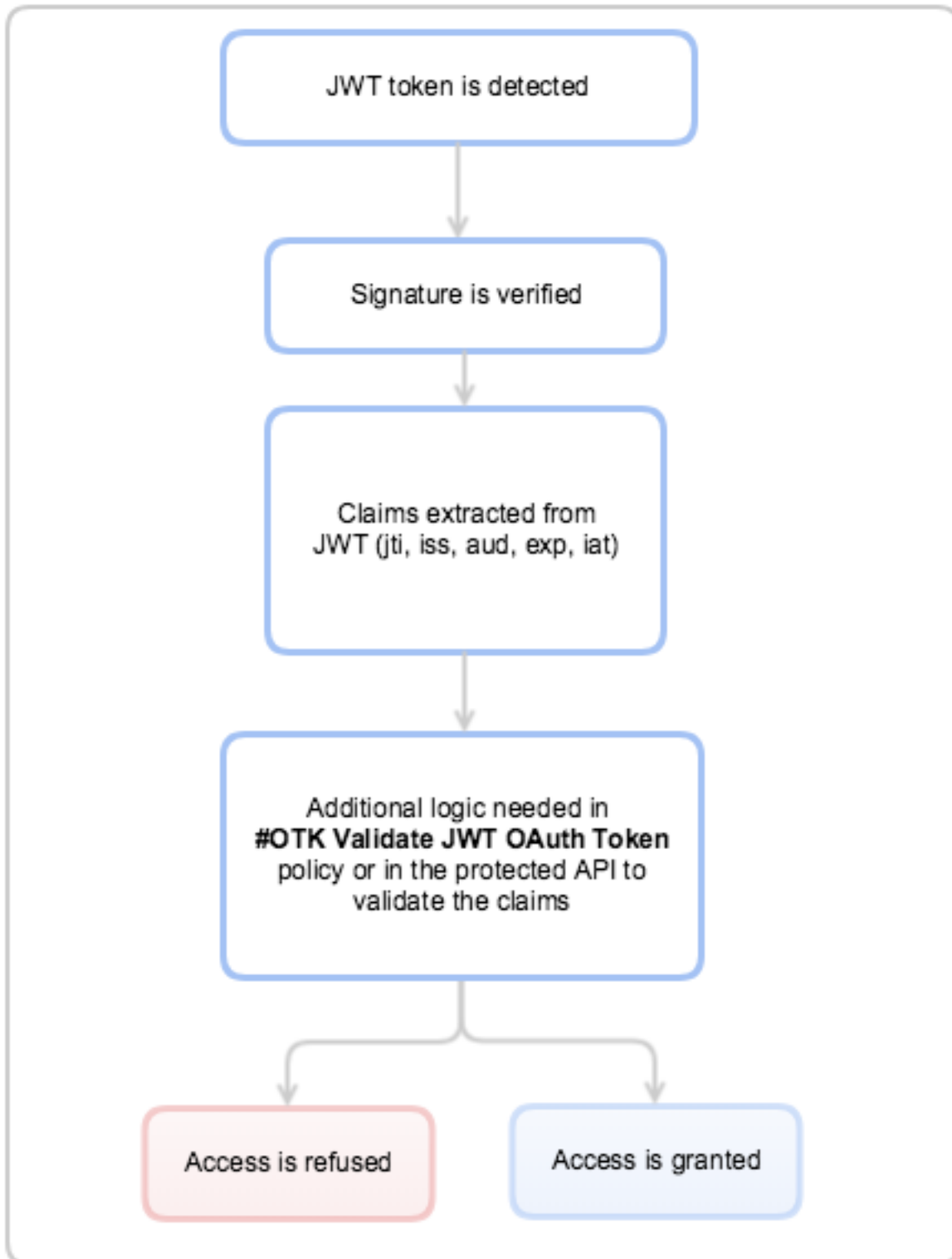
OTK validates an access token by first checking if it is a UUID or a JWT.

If a JWT is detected, the OTK verifies the signature, then extracts the jti from the JWT to validate it the same way as the UUID token.



Validate Without the Authorization Server Database

You can customize the validation behavior in the **#OTK Validate JWT OAuth Token** policy not to require the Authorization Server database query. By default, OTK supports validation of the signature only. To extract and validate additional information from the access token, such as iss, exp, aud, and scope, customize the validation.



Disable access to Authorization Server database and **Customize validation of JWT claims:**


1. Open the **#OTK Validate JWT OAuth Token** policy.

2. Enable the **Evaluate JSON Path Expression** assertions. Enabling extracts the listed values from the access token. Add logic to how the values should be validated by building your own policy.
3. Disable the **Set Context Variable query_db as String to: true** assertion.
4. **Save and Activate** the policy.


 Comment: Policy Fragment: #OTK Validate JWT OAuth Token

 Comment: Private key to decode JWT can be modified in the below assertion.

 Decode Json Web Token: validate using recipient key (Key: ssl)

 Set Context Variable payload_json as Message to: \${jwt.payload}

 \${payload_json}: Evaluate JSON Path Expression V2 – jti


 Comment: By default, the database will be queried for JWT validation (using 'JTI')

 Comment: Enable the following disabled assertions to not query the database when validating JWT

 \${payload_json}: Evaluate JSON Path Expression V2 – iss

 \${payload_json}: Evaluate JSON Path Expression V2 – aud

 \${payload_json}: Evaluate JSON Path Expression V2 – exp


 \${payload_json}: Evaluate JSON Path Expression V2 – token_details.scope

Enable to extract the values from the claims to be validated

▶  At least one assertion must evaluate to true

 Set Context Variable resp as Message to: <?xml version="1.0" encoding="UTF-8"?> <values xmlns="http...

 Comment: Use the following assertion to validate JWT by querying database for access_token (jti)

 Comment: Disable the following context variable assertion (query_db) if database query is not required for validation

 Set Context Variable query_db as String to: true

NOTE

If you do not use the Authorization server database for JWT validation, revoking the access token through OAuth Manager has no effect. The JWT access token remains valid until it expires.

Client Authentication

Client authentication is a method for a client to authenticate to the token endpoint. Use Client authentication in conjunction with some grant types to form a complete authentication request. For default grant types in OTK, see [Support Custom Grant Types](#). By default, a client is authenticated with a Client ID and a Shared Secret.

The authentication method that is used is based on what is registered during client registration. For OAuth client registration, see [Dynamic Registration](#).

Client Authentication Using JWT

JWT as client credentials is in general a more secure solution than a shared secret. All grant types that require Client to be authenticated are supported. Some advantages are:

- JWT can carry more information that helps server to verify the client.
- Replay attack protection prevents JWT to be used more than once.
- The private key JWT only stays on client side.
- Using jwks_uri allows client to rotate key without the need to update the new public key to the server.

When JWT is selected for client authentication, replay attack is performed to prevent a JWT being used more than once. The default check is done through cache for performance reasons and will not work for cluster environment. To disable the replay attack protection or to use a DB query for a JWT identity check, modify #OTK Replay Attack Protection policy.

Validation of JWT Client Credentials

The signature that is used to sign the JWT assertion and the following claims are always verified.

- iss: matches the identifier of the client.
- aud (audience): contains the authorization server token endpoint.
- sub: matches the identifier of the client.
- exp (expiry time): is greater than the current time.
- jti: will be cached until expiry time, and replay attack protection is performed

Create FIP Authentication for Dual Gateways

An X.509 FIP must be implemented to validate client certificates.

NOTE

In the dual-gateway scenario, perform all tasks described on this page on the Internal Gateway.

Before implementing a FIP, ensure that any needed certificates have been imported.

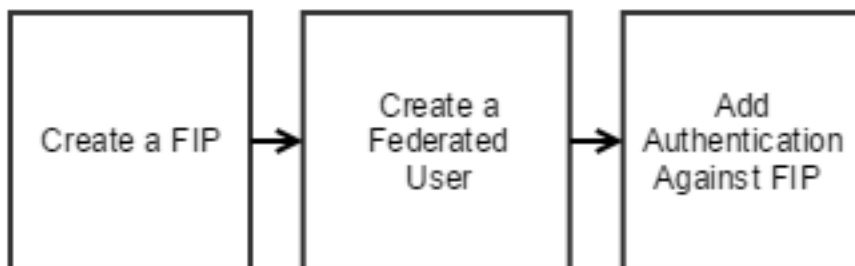
In Policy Manager, go to **Tasks, Certificates, Keys and Secrets, Manage Certificates**.

Certificates that must be imported include:

- The Gateway's own default SSL certificate
- The SSL certificate of any Gateway that is connecting as a client.

In the dual-gateway scenario, the Internal Gateway must import the SSL certificate of the External Gateway.

Figure 2: FIP-MAG

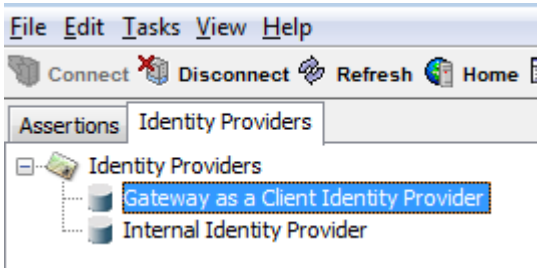


Create a FIP

To create a FIP:

1. Navigate to **Tasks, Identity Providers, Create Federated Identity Provider**.
2. Click **Provider Name** and type a name. For example: "Gateway as a Client Identity Provider".
3. For Credential Source Type Allowed, select only the **X.509 Certificate** checkbox. Leave the SAML Token checkbox unchecked.
Click **Next**.
4. Do not add any trusted certificates to this FIP. Leave the box blank.
Click **Next**. A warning box appears. Click **OK**.
5. For **Validation**, select **Validate Certificate Path**.
6. Click **Finish**.

Click the Identity Providers tab to verify that the FIP was created.

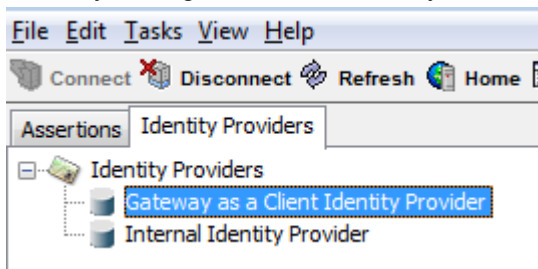


Create a Federated User

For each client connecting to the validation and storage endpoints (possibly including the Gateway itself), create a Federated User within this new FIP. Identify the client's certificate for its outbound TLS connection.

To create a federated user:

1. In Policy Manager, select the Identity Providers tab.



2. Right-click the "Gateway as a Client Identity Provider" FIP you created and select **Create User**. The Create Federated User dialog appears.
3. Click **X.509 Subject DN** and enter the complete DN of the client certificate that will be imported for this user. For example: CN=gateway.example.com
A default user name value is generated.
4. Select **Define Additional Properties** and click **Create**. The user properties dialog box appears.
5. Click the Certificate tab and click **Import**.
6. Import the SSL certificate of the client gateway as this user's certificate.
If the gateway is connecting to itself, select **Import from Private Key Certificate Chain** and choose the default SSL key.
If an external client (for example, a MAG in the DMZ) is connecting to these endpoints, select **Retrieve via SSL Connection (HTTP or LDAPS URL)**. Type a URL that leads to a listen port on the external client. For example: https://clientgateway.example.com:8443/
7. Click **Next**. View certificate details.
8. Click **Finish**.
If the user's subject DN is different from the one appearing in the imported certificate, a warning appears.
9. Click **OK** to close the properties window.

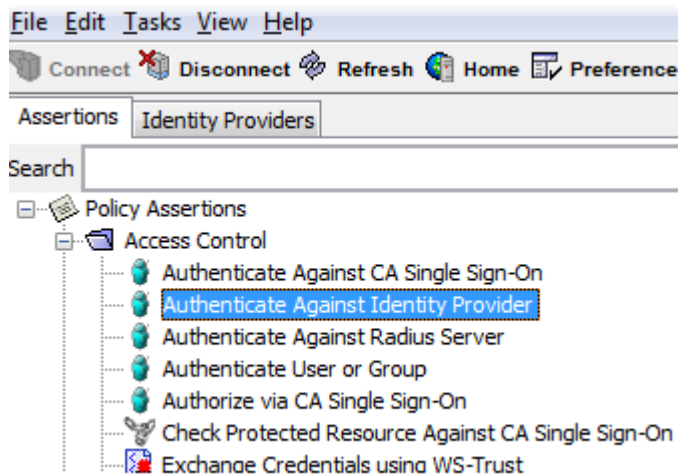
Add Authentication Against FIP

Add the authentication against the FIP you created to the **OTK FIP Client Authentication Extension**. This extension is included in the read-only OTK FIP Client Authentication policy.

To add authentication against a FIP:

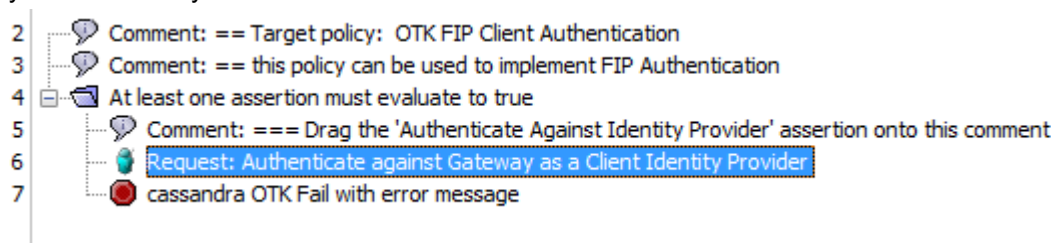
1. Go to OTK/Customizations/authentication/ and double-click the OTK FIP Client Authentication Extension. The extension policy opens in the right panel.
2. Enable the code block within the extension by right-clicking the disabled code and selecting **Enable Assertion**.

3. From the Access Control folder in the top left panel, locate the Authenticate Against Identity Provider



assertion

4. Drag the Authenticate Against Identity Provider assertion and drop it onto the associated Comment line in the OTK FIP Client Authentication Extension. A selection window appears when you drop the assertion. Select the Identity Provider you created as your FIP and click **OK**.



5. **Save and Activate.**

No further configuration is required. By default, the OAuth storage and validation endpoints use the OTK FIP Client Authentication encapsulated assertion.

Login and Consent Behavior

Client access to a protected resources includes two distinct actions:

- Providing credentials for login
- Providing user consent

The login action authenticates. The consent action grants authorization.

The following sections describe how to customize the default login and consent behaviour:

For an example of the default login and consent behaviour see, [Run the OAuth 2.0 Test Client](#).

See the [API documentation](#) for complete endpoint details.

Initial Request

The initial client request is directed to the `/auth/oauth/v2/authorize` endpoint.

Customize the request by including optional parameters in your request.

Required Parameters	Notes
client_id	The 'client_id' of the requesting client.

response_type	See Response Types below.
Optional Parameters	Notes
redirect_uri	<p>A 'redirect_uri' that was registered for this client. Required if multiple redirect_uri's have been registered for this client.</p> <p>If provided, only requests using a registered redirect_uri of this client are granted by the OAuth server. If the parameter is not included, the OAuth server uses the registered redirect_uri. If multiple redirect_uris have been registered, the request fails. If a redirect_uri is included and none was registered, the OAuth server uses the one included in the request</p>
scope	<p>Only scope values that were registered for this client are granted. If only non-matching scope values are requested, the request fails. For example:</p> <ul style="list-style-type: none"> scope=openid is required for openid requests. scope=msso is required for clients using the Mobile SDK.
nonce	Used to mitigate replay attacks. The authorization server rejects a second request if it has the same nonce value. Required when response_type=token id_token.
display	<p>Uses one of the following values:</p> <ul style="list-style-type: none"> page – displays an HTML web page. social_login – for display on mobile devices. Used with Mobile API Gateway only. Creates a JSON message response containing a list of social login providers.
prompt	<p>Uses one or more of the following values:</p> <ul style="list-style-type: none"> none (cannot be combined with other values) login consent <p>The prompt value indicates whether the server should prompt the user for login and/or consent. Default setting: prompt=login consent.</p> <p>Notes:</p> <p>If prompt=none. Indicates the server does not request user authentication or request consent as long as the user is currently logged in and the client has previously received requested grants. A cookie is used.</p> <p>If prompt=login, the authorization server must reauthenticate the user. The user is asked for credentials.</p> <p>If prompt=consent, the current user must have an active oAuth session using the same client and the same scope.</p>
id_token_hint	<p>Contains a previously issued id_token.</p> <p>If id_token_hint is included, then prompt=none must also be included.</p>
acr_values	<p>A space separated list of different values that indicates which Authentication Context Class Reference values are acceptable for the user authentication. Based on the requested acr claim value, the Authorization Server can set thresholds for allowing authentication, requesting re-authentication, or denying authentication. The value is forwarded to the /authorize/login API where the accepted values are defined and any thresholds can be set. Values appear in order of preference. The acr values should be compared against a configured acr_level. That acr_level is then used when creating an id_token.</p>

state	Value opaque to the server, used by the client to track its session. It will be returned as received.
-------	---

Response Types

response_type	Flow	Notes	Request Example	Response Notes
code	Authorization	<p>Clients using <code>response_type=code</code> are using a more secure method than token with regards to visibility of issued tokens. The flow involves multiple steps that are required between sending the initial request to receiving an <code>access_token</code>.</p>	<pre>GET /authorize? response_type=code &client_id=s6BhdRkqt3 &redirect_uri=https%3A%2F%2Fclient.example.org%2Fcb &scope=openid%20profile%20email &nonce=n-0S6_WzA2Mj &state=af0ifj5ldkj HTTP/1.1 Host: server.example.com</pre>	<p>The authorization code is issued from the authorization endpoint and exchanged for an access token from the token endpoint.</p>
token	Implicit	<p>Clients using <code>response_type=token</code> are considered to be 'public' clients and do not receive a <code>refresh_token</code>.</p>	<pre>GET /authorize? response_type=token &client_id=123abc34t2 &redirect_uri=https%3A%2F%2Fclient.example.org%2Fcb Host: server.example.com</pre>	<p>This type of client may be implemented in JavaScript. The client receives a login page that requests user credentials. On success, the client receives a token from the authorization endpoint.</p>
id_token	Implicit	<p>A successful response must include the parameter <code>id_token</code>. If a <code>redirect_uri</code> is supplied, the client is redirected after granting or denying access.</p>	<pre>GET /authorize? response_type=id_token &client_id=s6BhdRkqt3 &redirect_uri=https%3A%2F%2Fclient.example.org%2Fcb &scope=openid%20profile%20email &nonce=n-0S6_WzA2Mj &state=af0ifj5ldkj HTTP/1.1 Host: server.example.com</pre>	<p>The value of the <code>id_token</code> parameter is the ID Token, which is a signed JWT, containing three base64url encoded segments separated by period ('.') characters.</p>

code id_token	Hybrid	The 'code id_token' response_type must be used with scope=openid.	<pre>GET /authorize? response_type=code %20id_token &client_id=s6BhdRkqt3 &redirect_uri=https %3A%2F %2Fclient.example.org %2Fcb &scope=openid %20profile%20email &nonce=n-0S6_WzA2Mj &state=af0ifjsldkj HTTP/1.1 Host: server.example.com</pre>	The ID token is retrieved from the token access endpoint. Verifying and decoding the ID token exposes the claims.
token id_token	Implicit	The 'token id_token' response_type must be used with scope=openid.	<pre>GET /authorize? response_type=id_token %20token &client_id=s6BhdRkqt3 &redirect_uri=https %3A%2F %2Fclient.example.org %2Fcb &scope=openid %20profile%20email &nonce=n-0S6_WzA2Mj &state=af0ifjsldkj HTTP/1.1 Host: server.example.com</pre>	The client receives a login page that requests user credentials.

code token	Hybrid	A successful response must include an Access Token, an Access Token Type, and an Authorization Code.	<pre>GET /authorize? response_type=code %20token &client_id=s6BhdRkqt3 &redirect_uri=https %3A%2F %2Fclient.example.org %2Fcb &scope=openid %20profile%20email &nonce=n-0S6_WzA2Mj &state=af0ifjsldkj HTTP/1.1 Host: server.example.com</pre>	
code id_token token	Hybrid	The 'code id_token token' response_type must only be used with scope=openid. A successful response must include an Authorization Code, an id_token, an Access Token, and an Access Token Type.	<pre>GET /authorize? response_type=code %20id_token %20token &client_id=s6BhdRkqt3 &redirect_uri=https %3A%2F %2Fclient.example.org %2Fcb &scope=openid %20profile%20email &nonce=n-0S6_WzA2Mj &state=af0ifjsldkj HTTP/1.1 Host: server.example.com</pre>	
none		The Authorization Server should not return an OAuth 2.0 Authorization Code, Access Token, Access Token Type, or ID Token in a successful response to the grant request.		

Login Request

When prompt=login, the login request is sent to the **/auth/oauth/v2/authorize/login** endpoint.

This endpoint handles the following login actions:

- login
- cancel
- reset

The endpoint validates user credentials.

Contains the consent parameter which is none or active. Only active sessions can be granted at /authorize/consent.

Consent Request

When the consent parameter from the login endpoint is active, the consent request is sent to the **/auth/oauth/v2/authorize/consent** endpoint.

The API returns an HTML page for the user to grant or deny the request. Handles the grant or deny action requested from the consent page.

Success returns the following parameters within a URL fragment to the redirect_uri registered with the client:

Parameters	Notes
access_token	The issued OAuth 2.0 access token.
expires_in	The access_token lifetime in seconds
token_type	The OAuth 2.0 token type. Must be "Bearer".
scope	The granted scope values which may differ from the requested values.
id_token	The id_token (represented as JWT) associated with the authenticated session. Issued for response_type=token id_token if the requested scope includes openid.
id_token_type	The type of id_token. This is a OTK extension to allow the creation of other types. For example: SAML
state	Value opaque to the server, used by the client to track its session. Returned as received.

Grant Types

The following table describe requests for an access_token using a specified grant_type. For example: grant_type=authorization code

Grant Type	Notes
password	Use this grant_type value if the client was built by the enterprise that also implements the OAuth token server. Example: { "access_token":"115b8c ... 11a5", "token_type":"Bearer", "expires_in":3600, "refresh_token":"74b29d19-8b ... 7bb6bd1", "scope":"openid email" }
credentials	Use this grant_type value if the client is acting on its own behalf. No user consent is required. Use this grant_type value when response_type=token.

authorization code	Use this grant_type value when response_type=code is used and the client receives the authorization_code attached to a redirect URI. The client exchanges the authorization_code for an access_token. If the client includes "openid" as a scope value, additional OpenID Connect keys are included in the response.
urn:ietf:params:oauth:grant-type:jwt-bearer	Use this grant_type value when the client is in possession of an id_token (represented as JWT) of an authenticated user. Only id_tokens (JWT) that were issued by the OAuth server are accepted.
urn:ietf:params:oauth:grant-type:saml2-bearer	Use this grant_type value if the client is in possession of a SAML 2.0 token of an authenticated user. This scenario is useful in cases of federation where the SAML 2.0 token was signed by a trusted party.

For more information, see the following sections on each grant type:

grant_type=password

This grant_type can be used if the client was built by the enterprise that also implements the OAuth token server.

Request	
Method:	POST
Header:	content-type: application/x-www-form-urlencoded
Header:	authorization: Basic base64(client_id:client_secret) (This header can only be used if 'client_id' and 'client_secret' are NOT found within the message body and vice versa!)
Endpoint:	/auth/oauth/v2/token
Parameters:	grant_type=password&username=a-username&password=a-user-s-password&client_id=a-client_id&client_secret=a-client_secret&scope=a-list-of-scope-values
Optional:	scope: Only SCOPE values that have been registered for the client will be granted by the OAuth server

Response	
Header:	status: 200
Header:	content-type: application/json
Body:	Example: { "access_token":"115b8c ... 11a5", "token_type":"Bearer", "expires_in":3600, "refresh_token":"74b29d19-8b ... 7bb6bd1", "scope":"openid email" }

grant_type=client_credentials

This grant_type can be used if the client is acting on its own behalf. No user consent is required.

Request	
Method:	POST
Header:	content-type: application/x-www-form-urlencoded
Header:	authorization: Basic base64(client_id:client_secret) (This header can only be used if 'client_id' and 'client_secret' are NOT found within the message body and vice versa!)
Endpoint:	/auth/oauth/v2/token
Parameters:	Parameters: grant_type=client_credentials&client_id=a-client_id&client_secret=a-client_secret&scope=a-list-of-scope-values
Optional:	scope: Only SCOPE values that have been registered for the client will be granted by the OAuth server

Response	
Header:	status: 200
Header:	content-type: application/json
Body:	{ "access_token":"115b8c ... 11a5", "token_type":"Bearer", "expires_in":3600, "scope":"openid email" }

grant_type=authorization_code

Exchange the authorization_code for an access_token. A client has received the authorization_code attached to a redirect URI. The client now exchanges the authorization_code for an access_token by using grant_type 'authorization_code'.

Request	
Method:	POST
Header:	content-type: application/x-www-form-urlencoded
Header:	authorization: Basic base64(client_id:client_secret) (This header can only be used if 'client_id' and 'client_secret' are NOT found within the message body and vice versa!)
Endpoint:	/auth/oauth/v2/token
Parameters:	grant_type=authorization_code&code=the-received-authorization-code&client_id=a-client_id&client_secret=a-client_secret&redirect_uri
Optional:	redirect_uri: The value has to be included if it has been used in the initial request. It also has to match the original value

Response	
Header:	status: 200
Header:	content-type: application/json

Response	
Body:	{ "access_token":"115b8c ... 11a5", "token_type":"Bearer", "expires_in":3600, "refresh_token":"74b29d19-8b ... 7bb6bd1", "scope":"openid email" }

If the client included 'openid' as SCOPE in his request, additional keys are included in the response:

```
... "id_token":"eyJ0eXAiOi1v8 ... JZu_LsN851Vtfc5pcIqJc", "id_token_type":"urn:ietf:params:oauth:grant-type:jwt-bearer" ...
```

The id_token (JWT) can be used with grant_type=urn:ietf:params:oauth:grant-type:jwt-bearer.

grant_type=urn:ietf:params:oauth:grant-type:jwt-bearer

This grant_type can be used if the client is in possession of an id_token (represented as JWT) of an authenticated user. Only id_token (JWT) that were issued by the OAuth server are accepted.

Request	
Method:	POST
Header:	content-type: application/x-www-form-urlencoded
Header:	authorization: Basic base64(client_id:client_secret) (This header can only be used if 'client_id' and 'client_secret' are NOT found within the message body and vice versa!)
Endpoint:	/auth/oauth/v2/token
Parameters:	grant_type=urn%3Aietf%3Aparams%3Aoauth%3Agrant-type%3Ajwt-bearer&assertion=a-jwt&client_id=a-client_id&client_secret=a-client_secret&scope=a-list-of-scope-values
Optional:	scope: Only SCOPE values that have been registered for the client will be granted by the OAuth server

Response	
Header:	status: 200
Header:	content-type: application/json
Body:	{ "access_token":"115b8c ... 11a5", "token_type":"Bearer", "expires_in":3600, "refresh_token":"74b29d19-8b ... 7bb6bd1", "scope":"openid email" }

grant_type=urn:ietf:params:oauth:grant-type:saml2-bearer

This grant_type can be used if the client is in possession of a SAML 2.0 token of an authenticated user. This scenario is useful in cases of federation where the SAML 2.0 token was signed by a trusted party.

Request	
Method:	POST
Header:	content-type: application/x-www-form-urlencoded
Header:	authorization: Basic base64(client_id:client_secret) (This header can only be used if 'client_id' and 'client_secret' are NOT found within the message body and vice versa!)

Request	
Endpoint:	/auth/oauth/v2/token
Parameters:	grant_type=urn%3Aietf%3Aparams%3Aoauth%3Agrant-type%3Asaml2-bearer&assertion=a-base64-encoded-saml-token&client_id=a-client_id&client_secret=a-client_secret&scope=a-list-of-scope-values
Optional:	scope: Only SCOPE values that have been registered for the client will be granted by the OAuth server

Response	
Header:	status: 200
Header:	content-type: application/json
Body:	{ "access_token":"115b8c ... 11a5", "token_type":"Bearer", "expires_in":3600, "scope":"openid email" }

grant_type=refresh_token

This grant_type can be used if the client is in possession of a refresh_token. The request will only be successful if the refresh_token has not expired. The parameter 'SCOPE' can only include the same or a subset of values that were originally requested. The refresh_token can only be use once.

Request	
Method:	POST
Header:	content-type: application/x-www-form-urlencoded
Header:	authorization: Basic base64(client_id:client_secret) (This header can only be used if 'client_id' and 'client_secret' are NOT found within the message body and vice versa!)
Endpoint:	/auth/oauth/v2/token
Parameters:	Parameters: grant_type=refresh_token&refresh_token=a-refresh-token&client_id=a-client_id&client_secret=a-client_secret&scope=a-list-of-scope-values
Optional:	scope: Only SCOPE values that have been registered for the client will be granted by the OAuth server

Response	
Header:	status: 200
Header:	content-type: application/json
Body:	{ "access_token":"115b8c ... 11a5", "token_type":"Bearer", "expires_in":3600, "refresh_token":"74b29d19-8b ... 7bb6bd1", "scope":"openid email" }

Customizing Authorization Server Settings

See [Configure the Authorization Server](#).

Multiple Session Support

Describes how to configure OAuth policies to support multiple sessions for a single resource owner/client id combination.

Business Value

Mobile app developers want to run multiple instances of their apps to provide a better experience for end users by:

- Allowing users to run an app from different tabs in their Web browser
- Persisting data in the previous session
- Using the same user credentials to access a previous session

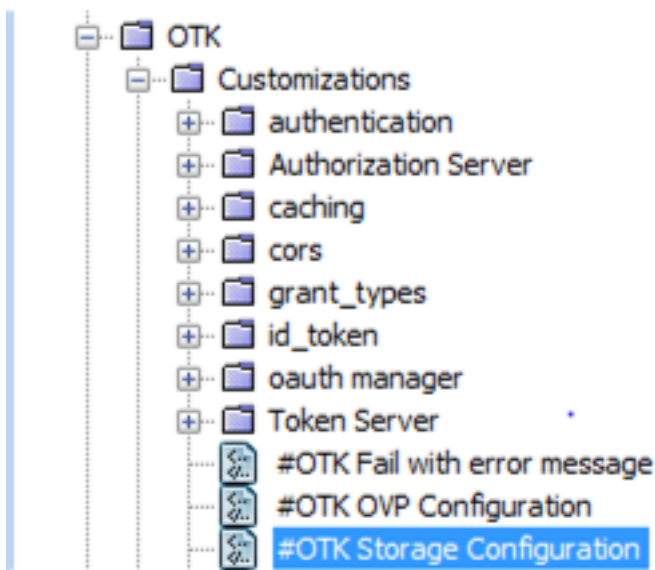
Restrictions

- Only OAuth 2.0 token flows support multiple app instance functionality
- OAuth 1.0 token flows (**/auth/oauth/v1/token**) still work, but multiple app instance functionality is not available

Configure Multiple Session Support

To configure multiple session support:

1. Access the customizable **#OTK Storage Configuration** policy:



2. Configure the following context variables.

Context Variable	Description	Values
max_oauth_token_count	Maximum number of OAuth tokens per app and resource owner combination.	Default: 1 Options: String Note: Values between 5-10 are suitable for most implementations. If you use higher values, performance can be affected.
max_token_behaviour	Action when the number of OAuth tokens per app and resource owner combination is reached.	Default: cycle Options: <ul style="list-style-type: none"> • cycle = Grant a new token, but remove the oldest token. • error = Return an error ("Max number of tokens reached"), and deny further tokens.

By default, the `max_oauth_token_count` is set to 1, indicating only a single session is allowed for the same resource owner/client id combination. To allow for multiple sessions by the same resource owner/client id combination, increase the `max_oauth_token_count` value. When the set value is reached, no additional sessions can be started.

For example, to allow an app in a Java script client to be opened across multiple browser tabs, set a custom value for the `max_oauth_token_count`.

In OTK versions before 4.0, only one session was permitted for the same resource owner/client id combination.

High Volume Issue

A high volume of concurrent requests **from the same resource owner/client** can cause the `max_oauth_token_count` restriction to be ignored. If you encounter this behavior, and consider the request volume acceptable, adjust the configured value to better accommodate the concurrent request load. Run the database clean up task to remove unused persisted tokens. Query the number of sessions in the database to identify any misuse (deliberate request flooding).

Solutions to mitigate the issue with a high volume of concurrent requests:

- Design apps that do not request multiple concurrent sessions from the same resource owner/client.
- Test your app and check the database for the number of concurrent sessions. If the configured number of acceptable sessions is exceeded, adjust the `max_oauth_token` count accordingly.
- Configure the OTK to accept an infinite number of sessions. Clean up the database. Query the database for maximum number of sessions to identify peaks that may be an indicator for misuse.

Support Custom Grant Types

OAuth Tool Kit lets you grant limited access to resources to a third party without exposing the credentials. Grant types are various methods to obtain access tokens and to allow different types of access to resources.

Each of the following default grant types has a corresponding policy found in OTK/Policy Fragments/`grant_types`:

- authorization_code
- client_credentials
- jwt-bearer
- password
- refresh_token
- SAML

NOTE

For information on customizing the SAML grant type, see [Support the SAML Grant Type](#).

The OAuth 2.0 specification also supports **custom** grant types.

To support a custom grant type, the following tasks are required:

Create the Custom Grant Type

To create an OAuth 2.0 custom grant type:

1. Open the **OTK grant_type=CUSTOM** extension policy found in OTK/Customizations/grant_types. The policy provides an example implementation for setting up grantTypeCustom1 and grantTypeCustom2. The target policy for this extension is located at the /auth/oauth/v2/token endpoint.
2. Double-click the Compare Variable assertion for grantTypeCustom1.
3. Select the rule, click **Edit**, and type the name of your custom grant type. For example, "phone". Use the check box to indicate whether the name is case-sensitive.
4. Click **OK**.
The next steps persist the cache contents in the database.
5. From the upper panel of the Policy Manager, under the Assertions tab, type OTK Token Storage.

- ↴ OTK Token Storage (access_token)
- ↴ OTK Token Storage (access_token, refresh_token)
- ↴ OTK Token Storage (oauth_token)
- ↴ OTK Token Storage (temporary token)

Insert one of the assertions into the OTK grant_type=CUSTOM extension policy.

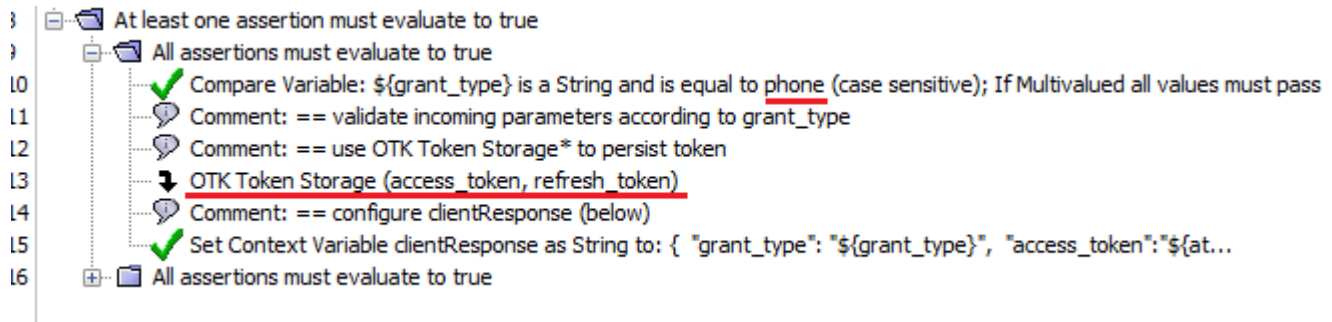
Assertion	Notes
OTK Token Storage (access_token)	Persists an OAuth 2.0 access_token. The token secret is optional.
OTK Token Storage (access_token, refresh_token)	Persists an OAuth 2.0 access_token and refresh_token. The token secret is optional.
OTK Token Storage (oauth_token)	Persist OAuth 1.0 token. The request_token is exchanged for the oauth_token.
OTK Token Storage (temporary_token)	Persists temporary token (authorization_code, request_token). Token Secret and Resource Owner are optional.

6. Configure the properties for the selected OTK Token Storage assertion. The following table shows some default configured properties for the access_token, refresh_token, and temporary_token.

OTK Token Storage Property	Variable	Notes
Redirect URI	\${redirect_uri}	temporary_token property only

Client	\${client}	refresh_token property only
Client Key	\${client_id}	
Client Name	\${client_name}	
Expiration	\${at_expiration}	
Resource Owner	\${client_name}	
Refresh Token Expiration	\${reexpiration}	refresh_token property only
Refresh Token	\${rtoken}	refresh_token property only
SCOPE	\${scope.granted}	
Token Status	ENABLED	
Access Token	\${at_token}	
Token Secret*	\${token_secret}	
JSON with session values	\${session_output}	

7. After configuring the properties to store for your custom token, click **OK**.
The policy is updated with the custom grant type name and the OTK Token Storage assertion.



8. Perform any additional customization such as:
- Adding a custom scope value
 - Providing validation logic
9. Edit the clientResponse string for each custom grant type. By default the clientResponse is set to:

```
{
  "grant_type": "${grant_type}",
  "access_token": "${at_token}",
  "token_type": "Bearer",
  "expires_in": ${at_lifetime},
  "refresh_token": "${rtoken}",
  "scope": "${scope.granted}"
}
```

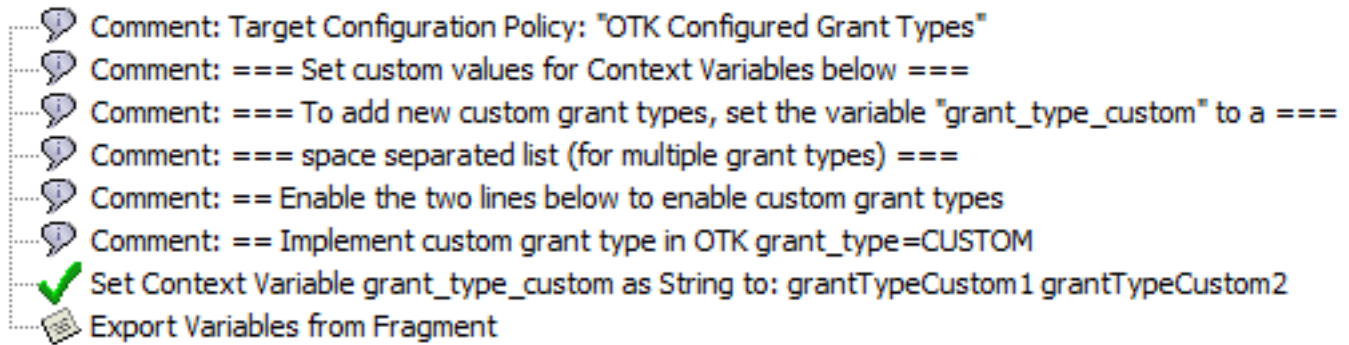
10. When your customization is complete, **Save and Activate**.

Enable the Custom Grant Type

After the custom grant type is configured, enable it in the **#OTK Configured Grant Types** policy. You can enable multiple custom grant types.

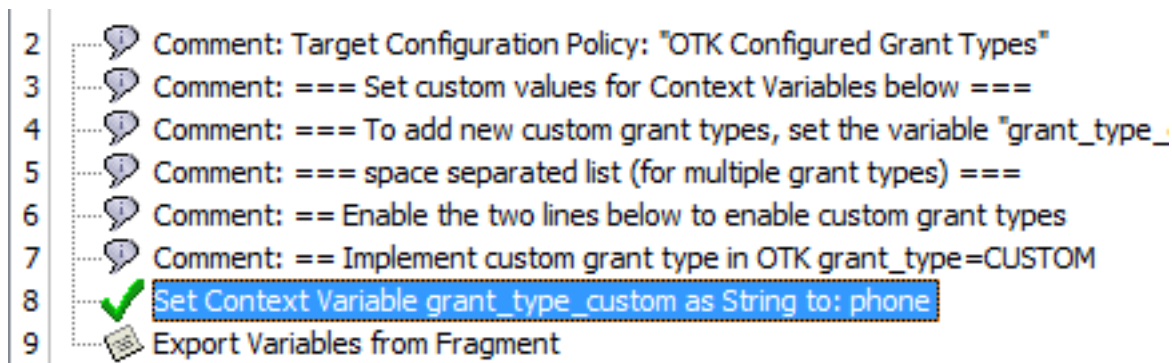
To enable the custom grant type:

1. Open the **#OTK Configured Grant Types** policy found in **OTK/Customizations/grant_types**.
2. Enable the final two assertions. These assertions set the `grant_type_custom` Context Variable and export the variable.



The default code includes support of two custom grant types, each separated by a space.

3. Double-click the first Set Context Variable grant_type_custom assertion. The Compare Expression Properties dialog appears.
4. Select the existing Rule and click **Edit**.
5. Remove the default grant type name and type the name of the custom grant type in the OTK grant_type=CUSTOM policy.
6. For example, "phone". Click **OK**.



7. **Save and Activate.**

Support Optional Authentication Mechanisms

For user authentication or any other type of authentication, you can integrate OTK with various Identity Providers. The policy responsible for user authentication is OTK User Authentication that is located at OTK/Policy Fragments/ authentication. By default, username password authentication is supported. However, you can configure support for authenticating against external directories such as LDAP or CA Single Sign-On (formerly known as SiteMinder®). Configuration includes providing connector details and customizing policy to refer to the custom identity provider.

The following tasks that are related to optional authentication support are described:

Create a Custom Identity Provider

Creating a Custom Identity Provider allows you to authenticate requests with an existing external server, such as an LDAP server.

NOTE

An LDAP Identity Provider is only a connector to an existing LDAP directory. The Policy Manager cannot be used to create, edit, or delete LDAP Identity Provider users or groups. To perform such tasks, use the tools that are provided with your LDAP directory.

To create a Custom Identity Provider:

1. In the Policy Manager toolbar, navigate to Tasks, Identity Providers.
2. Select the type of Identity Provider connection to create.
3. Follow the creation wizard instructions to configure connection settings for the Identity Provider.

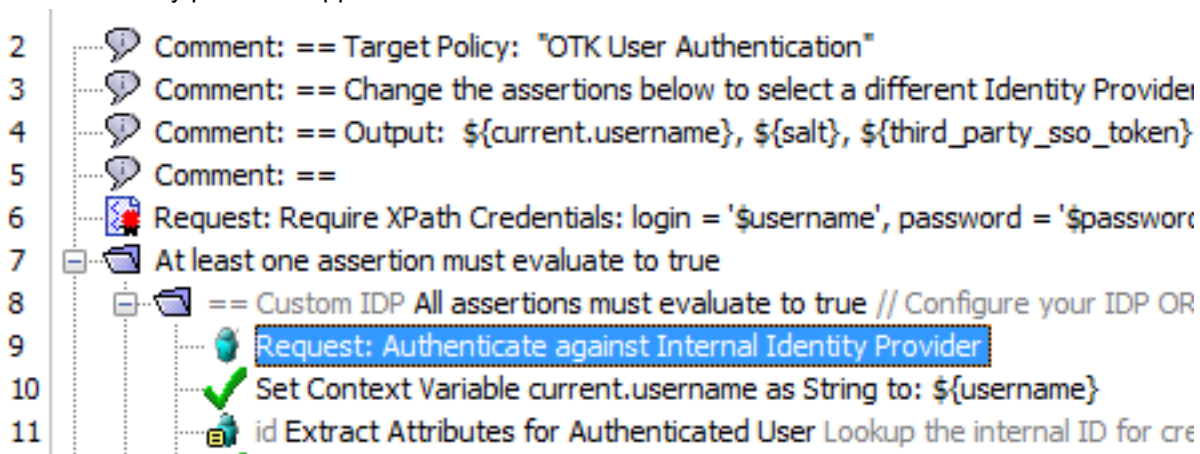
For more information, see the "LDAP Identity Providers" topic on the [CA API Gateway documentation site](#).

Authenticate against a Custom Identity Provider

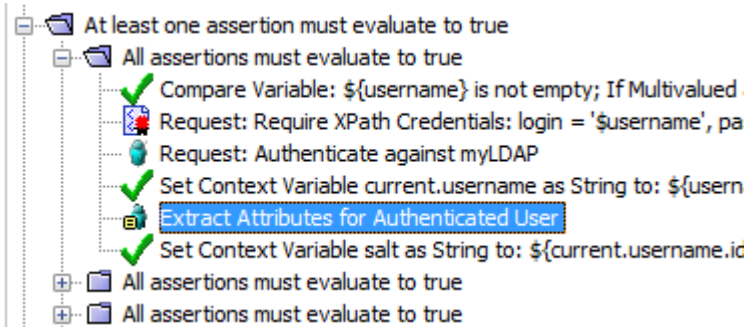
By default, OTK provides the Internal Identity Provider (IIP). The OTK User Attribute Lookup Extension is hard-coded with default IIP and should only be used in test environments, not in production. Customize this policy to integrate OTK with an Identity Provider of your choice.

To authenticate against a custom identity provider:

1. Open the **OTK User Authentication Extension** policy in the Customizations folder. Expand folders and show comments.
2. Search for the `==Custom IDP` comment.
3. Double-click the "Request Authenticate against Internal Identity Provider" assertion.
A list of identity providers appears.



4. Select the identity provider that you created then select **OK**.
5. In the same section of code, double-click the **Extract Attributes for Authenticated User** assertion and select the same Custom Identity Provider.



6. Click **OK**.
7. **Save and Activate** the policy.

NOTE

Although the LDAP identity provider stores usernames as **case insensitive**, MAS Storage database retrieval is **case sensitive** with regards to the authenticated username and key retrieval. As a result, you can successfully log into your cloud storage, but you can access your data only if the username meets the case-sensitive requirements of the MAS Storage database.

Consider a policy of creating all usernames in lowercase and treating them as case sensitive.

Authenticate against CA SiteMinder

You must have an existing SiteMinder installation running and configured to work with the CA API Gateway. For configuration details, refer to SiteMinder information in the CA API Gateway documentation.

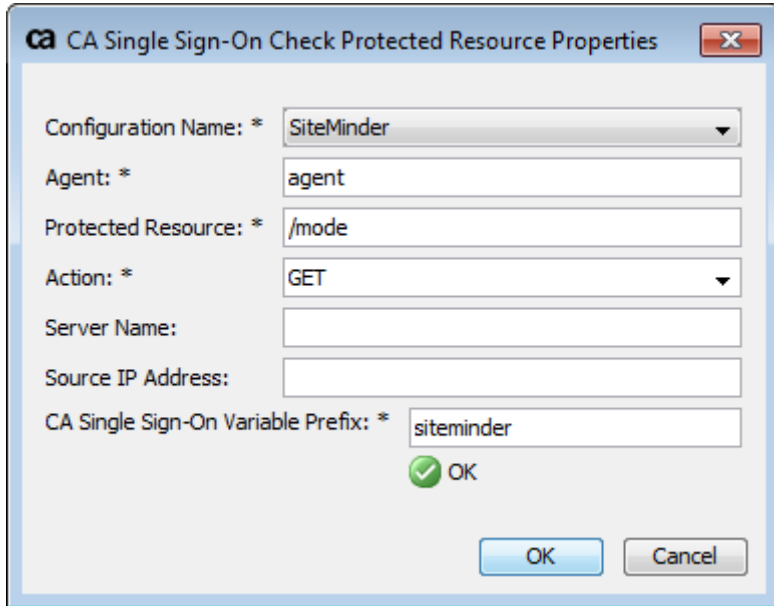
After performing any of the following customization tasks, remember to **Save and Activate** the modified policies.

Provide Existing SiteMinder Configuration Properties

The property values that are entered must match the configuration of your existing SiteMinder installation.

To provide the existing SiteMinder configuration properties:

1. In the OTK/Customizations/authentication folder, open the **#OTK SiteMinder Check Protected Resource** policy.
2. Double-click the "Request: Check Protected Resource Against CA Single Sign-On" assertion.
3. Configure the properties.
The following screen shot contains example values only. Modify these values to match your registered SiteMinder Configuration properties.

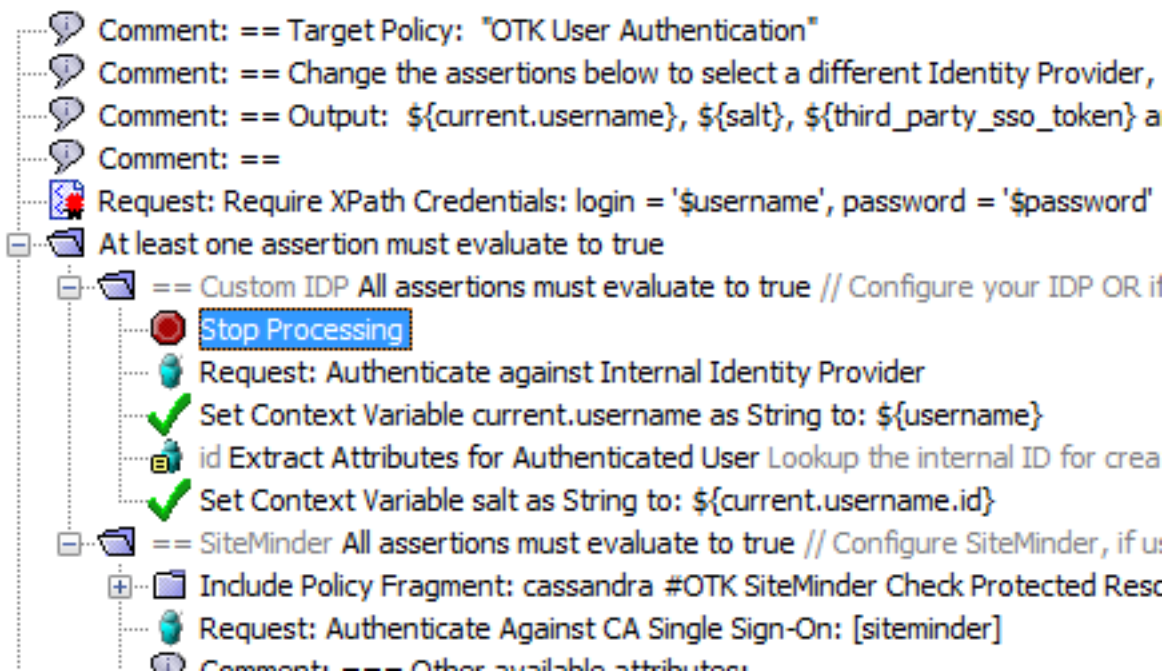


Enable SiteMinder Authentication

By default, the SiteMinder Authentication section of the extension policy is effectively disabled by a Stop Processing assertion.

To enable and customize SiteMinder Authentication:

1. In the OTK/Customizations/authentication folder, open the **OTK User Authentication Extension** policy.
2. Expand all folders. Show Comments.
3. Search for the comment: `==SiteMinder`.
4. Because you are not using a Custom IDP, drag the "Stop Processing" Assertion from the SiteMinder section to the top of the Custom IDP section.



Select Authentication Credentials

To set which type of credentials are used for authentication (user authentication, or device authentication):

1. Open the **OTK User Authentication Extension** policy. Expand all folders. Show Comments.
2. Double-click the "Request Authenticate Against CA Single Sign-On: [siteminder]" assertion.
3. Set any of the properties.

Authenticate Against CA Single Sign-On Properties

CA Single Sign-On Variable Prefix: *

OK

Credentials

Use Last Credentials

Specify Credentials

Supported Credential Types

Username Password Username:

X.509 Certificate Certificate CN or DN:

Create SSO Token

Use SSO Token from Context Variable:

Customize Even More

Additional customization available in the OTK User Authentication Extension policy can be performed for **username**, **salt**, and **outputs**.

By default, the **username** context variable string is set as: `${siteminder.smcontext.attributes.ATTR_USERNAME}`.

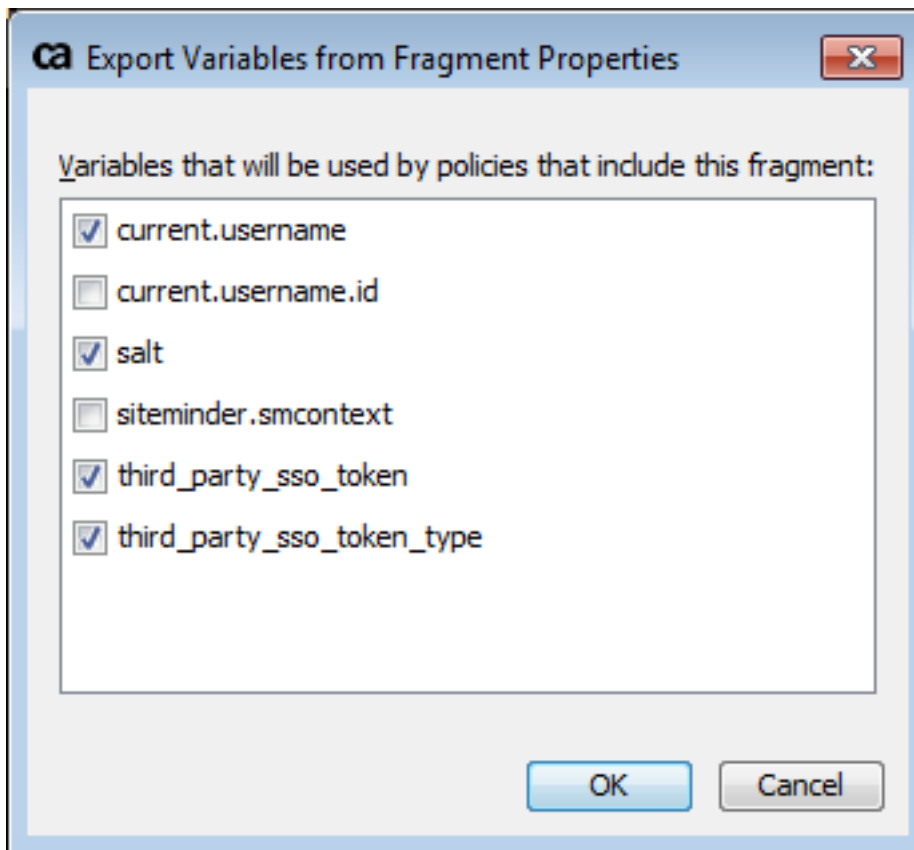
You can set `${current.username}`, by replacing `ATTR_USERNAME` with any of the SiteMinder variables shown in the following table.

Variable	Notes
ATTR_USERDN	The user's distinguished name as recognized by SiteMinder.
ATTR_USERUNIVERSALID	The user's universal ID. It could be the name from the LDAP.
ATTR_AUTH_DIR_OID	The object ID of the directory where the user has been authenticated.
ATTR_AUTH_DIR_NAME	The name specification of the directory where the user has been authenticated
ATTR_AUTH_DIR_SERVER	The server specification of the directory where the user has been authenticated
ATTR_AUTH_DIR_NAMESPACE	The namespace specification of the directory where the user has been authenticated.
ATTR_USERNAME	The user's display name.

By default, the **salt context variable** is set as: `${siteminder.smcontext.attributes.ATTR_USERUNIVERSALID}`

By default the **outputs** from this extension are: `${current.username}`, `${salt}`, `${third_party_sso_token}` and `${third_party_sso_token_type}`.

Click the "Export Variables from Fragment" assertion to modify output variables.



Support the SAML Grant Type

By default, support for the SAML token grant type is disabled in the policies delivered in the OAuth Toolkit.

The following tasks are required only if you intend to support the SAML grant type:

NOTE

CA Mobile API Gateway installations: In a dual MAG scenario, perform the following tasks on the Gateway in the DMZ.

Select SAML Options for SSL Certificates

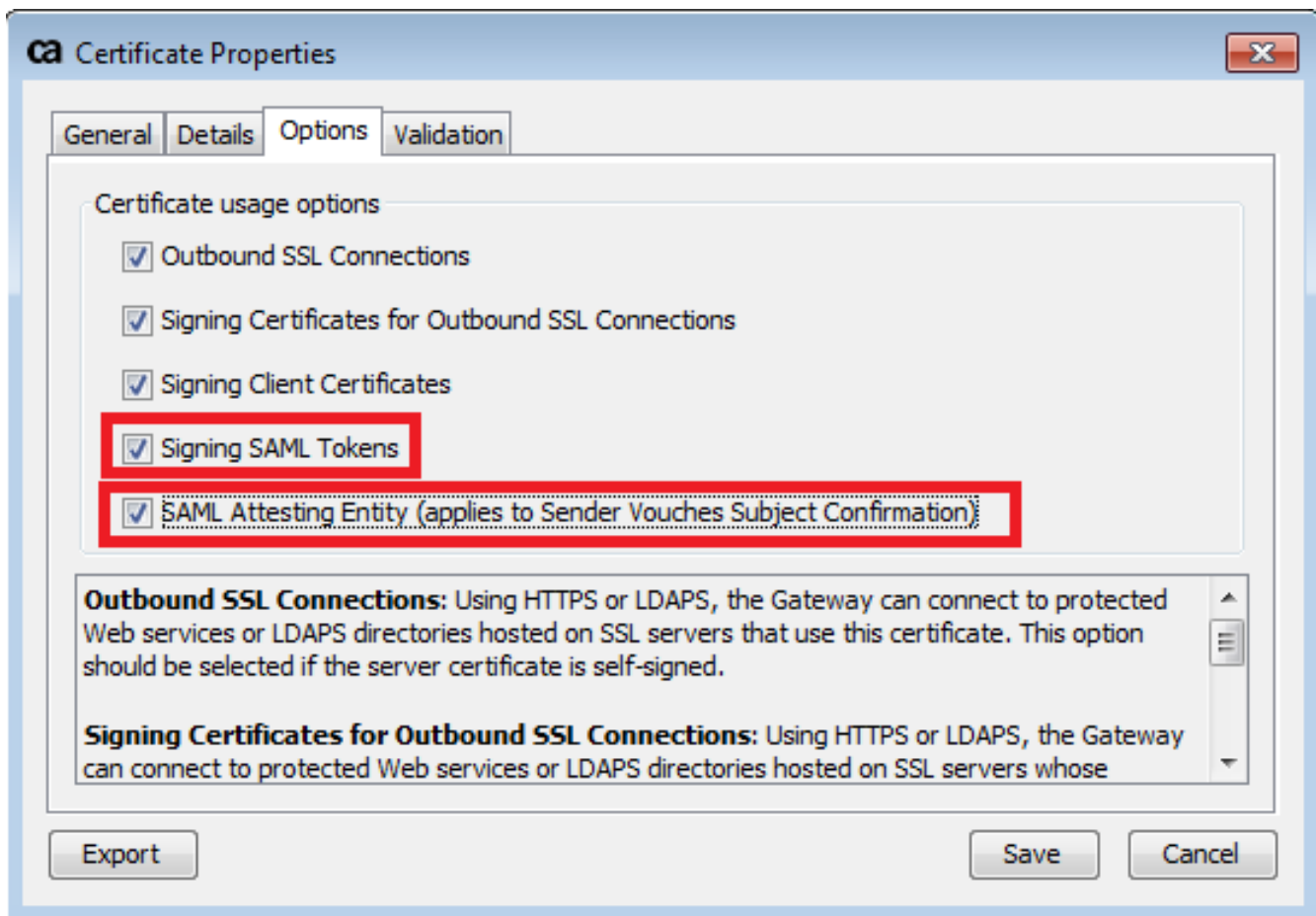
Certificates imported into the FIP include:

- The default certificate of the Gateway.
- The SSL certificate of any Gateway that is connecting as a client.

These certificates must have the additional options for SAML checked.

To select SAML options for SSL certificates:

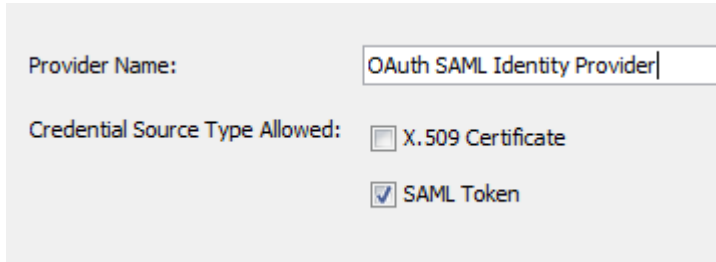
1. In Policy Manager, go to **Tasks, Certificates, Keys and Secrets, Manage Certificates**.
2. Double-click the certificate to view properties. Select the Options tab.
3. Ensure that the two SAML options are checked. If they are not, select them and click **Save**.



Create a FIP

To create a FIP:

1. Navigate to **Tasks, Identity Providers, Create Federated Identity Provider**.
2. Click **Provider Name** and type "OAuth SAML Identity Provider".
3. For **Credential Source Type Allowed** select only the **SAML Token** checkbox. Leave the X.509 Certificate checkbox unchecked.



Provider Name:

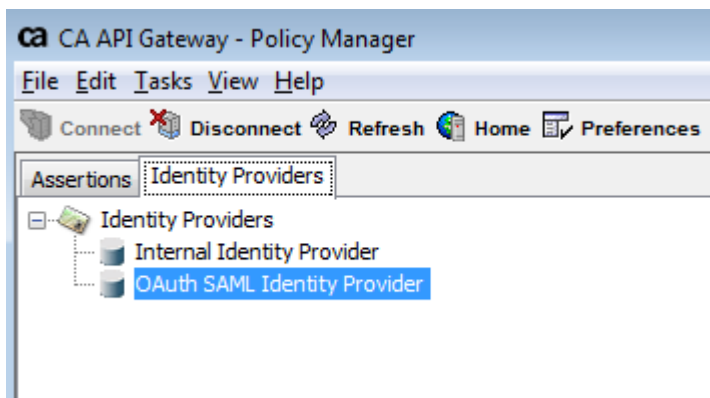
Credential Source Type Allowed:

X.509 Certificate

SAML Token

4. Click **Next** to add trusted certificates.
Trusted certificates include the gateways own default SSL certificate, and the certificate of any Gateway that is connecting as a client.
5. Click **Add**. Click **Search**.
A list of certificates appears. Click the gateway's own default SSL certificate. To add any additional certificates, use Ctrl-click.
Click **Select**.
The certificates appear on the Trusted Certificates list.
Click **Next** to set certificate validation options.
6. For **Validation**, select **Validate Certificate Path**.
7. Click **Finish**.

Verify the FIP was created by clicking the Identity Providers tab in the upper left panel of the Policy Manager.



Now enable the SAML grant type.

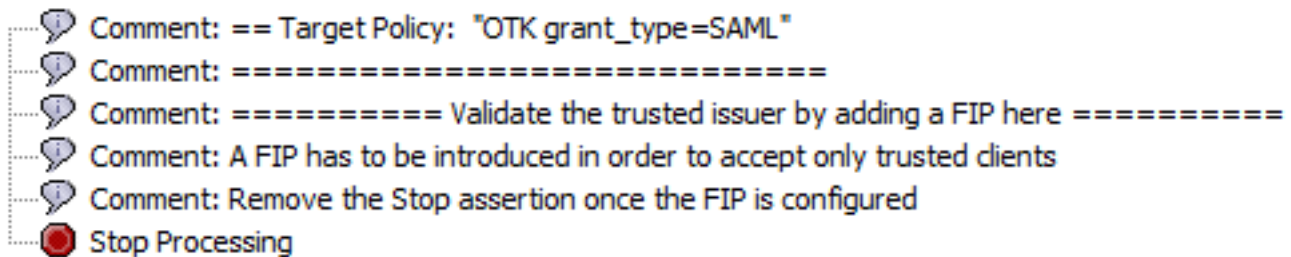
Enable the SAML Grant Type

This task is required if you intend to support the SAML grant type. Before you can enable the SAML grant type, you must create a FIP.

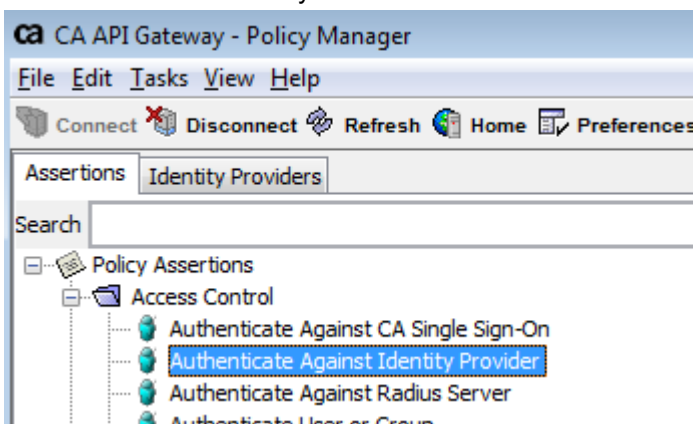
The following instructions add validation of the SAML Token Signer to SAML Token Grant Type policies. The SAML grant type is not supported until the Stop assertion is replaced with an Authenticate Against Identity Provider assertion. The FIP is selected as the provider.

To enable the SAML grant type:

1. Navigate to OTK/Customizations/grant_types and open the **OTK grant_type=SAML Authentication** policy.
2. Locate the Stop Processing assertion.



3. Replace the Stop Processing assertion in the policy with an **Authenticate Against Identity Provider** assertion. Find the assertion under Policy Assertions/Access Control.



Drag the Authenticate Against Identity Provider assertion and drop it directly on the Stop Processing assertion.

4. A "Change Authentication Identity Provider" dialog box asks you to select an Identity Provider. Select the **OAuth SAML Identity Provider** FIP you created. Click **OK**.
5. Right-click the Request: Authenticate against OAuth SAML Identity Provider. Choose **Select Target Message**. Select **Other Context Variable** and type "bearerToken". Click **OK**.
6. Verify that the assertion label is now:
\${bearerToken}: Authenticate against OAuth SAML Identity Provider
7. Disable or delete the Stop Processing assertion.
8. **Save and Activate** the policy.

Verify the Installation

Verify the following components of your OAuth Tool Kit:

The test clients are used to verify installation changes and to access secured API endpoints of platforms.

Note the following security precautions when using the test clients:

- Do not install the test client on product systems.
- Do not install the test client on a Gateway that is available on the Internet.
- Modify the test client to use your own specific client credentials.
- Remove the test client from the OAuth Manager when it is no longer needed.

Set the Callback URL for Test Clients

OPTIONAL

This task is required if you want to use test clients to verify the installation. It is recommended you perform this task.

To run the test clients, you must first configure the callback URL through the OAuth Manager.

Set the callback URL for each of the following test clients:

- OAuth2Client
- OpenID Connect Basic Client Profile
- OpenID Connect Implicit Client Profile

The name of the client appears in the third column from the left.

To set the callback URL:

1. In a browser, open the OAuth Manager by navigating to the following URL: `https://<hostname>:8443/<iModifier>/oauth/manager`
2. Log in with an administrator account.
3. Click **Clients**.
4. Locate the test client you want to use. For example, OAuth2Client.

9	TestClient1.0	OAuth1Client	oob	OAuth 1.0 test client hosted on the ssg	Layer7 Technologies Inc.	admin	0	0	DELETE	EDIT	LIST KEYS
10	TestClient2.0	OAuth2Client	confidential	OAuth 2.0 test client hosted on the ssg	Layer7 Technologies Inc.	admin	0	0	DELETE	EDIT	LIST KEYS
11	123456800-0sk	OpenID Connect	confidential	Test for OpenID	Layer7	admin	0	0	DELETE	EDIT	LIST KEYS

5. At the far right of the table row, click **List Keys**.
The List Client Keys page appears.
6. For the selected client, at the far right of the table row, click **Edit**. You may need to scroll horizontally to access the button.
The Edit Client Key page appears.
7. Replace the **Callback URL** value with the protocol, hostname, port, and optional instance modifier of your gateway. There are two URLs to edit.
8. Is the `auth=done` key value pair at the end of both URLs? If not, add it.
In the following Callback URL example, `server2` is the instance modifier and `myGateway.com` is the hostname.
`https://myGateway.com:8443/server2/oauth/v2/client/authcode?auth=done,https://myGateway.com:8443/server2/oauth/v2/client/implicit?auth=done`
9. Click **Save**.
10. Click **Clients** to return to the client list.

Repeat this task for any of the other test clients you intend to use.

NOTE

When you click Clients, no clients are listed? The most common reasons are:

No clients are registered for the username. To see the test clients, sign in as a user with the administrator role.

The OTK is integrated with the CA API Portal. In this case, clients are managed in the API Portal, not the OAuth Manager. For more information, see [Manage API Keys with CA API Portal](#).

Run the OAuth 2.0 Test Client

The test client is used to verify installation changes and to access OAuth 2.0-secured API endpoints of platforms. This section describes how it works and how it can be configured.

Run the Client

Have you set the callback URL for the test client? See [Post-Installation Tasks](#).

WARNING

Note the following security precautions when using the test client:

- – Do not install the test client on production systems.
- Do not install the test client on a Gateway that is available on the Internet.
- Modify the test client to use your own specific client credentials.
- Remove the test client from the OAuth Manager when it is no longer needed.

To run the OAuth 2.0 test client:

1. Navigate to the following URL in a browser:
`https://<Gateway_host>:8443/<InstanceModifier>/oauth/v2/client`
The OAuth Client Test Application screen is displayed.
2. Navigate between the other OAuth 2.0 Test Clients:
3.
 - Authorization Code
 - Implicit
 - Resource Owner Password Credentials
 - Client Credentials
 - SAML Bearer

Each OAuth 2.0 Test Client tests its own grant type. If you are only using a subset of the available OAuth grant types, you can ignore the other test clients.

Each client app maintains its own token. Each time you initiate a new OAuth session, the current access token is overwritten.

The access token in memory is used to call the test API. In the case of SAML, a SAML token is also maintained in memory and overwritten each time you initiate a new one.

Grant type: Authorization code

Current Access Token:

Initiate new OAuth handshake

INITIATE

Refresh Current Access Token

REFRESH

Call API Using Current Access Token

Target:

CALL API

Get an Access Token

To get an access token before calling an API:

1. Click the OAuth V2 Clients on the top bar.
2. Click any of the test clients identified by grant type on the black bar.
3. Click **Initiate**.
The OAuth 2.0 Authorization Server login page is displayed. (The Resource Owner Password Credentials client requires this step before clicking Initiate).
4. Enter your credentials.



OAuth 2.0 Authorization Server

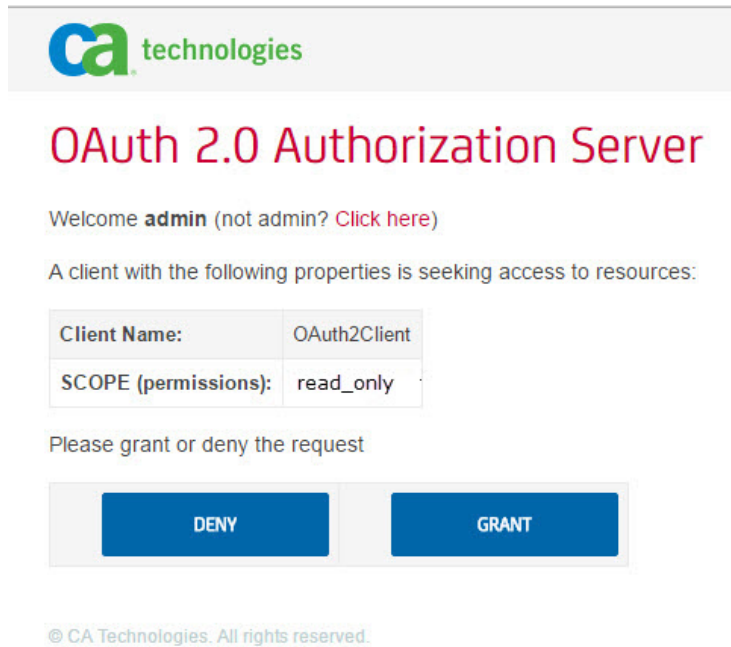
Please login:

Username *

Password *

NOTE

Log in with social login credentials is only available for clients when CA Mobile API Gateway is installed. The authorization page appears displaying the requested scope. You are asked to **Grant** or **Deny** the request.



ca technologies

OAuth 2.0 Authorization Server

Welcome **admin** (not admin? [Click here](#))

A client with the following properties is seeking access to resources:

Client Name:	OAuth2Client
SCOPE (permissions):	read_only

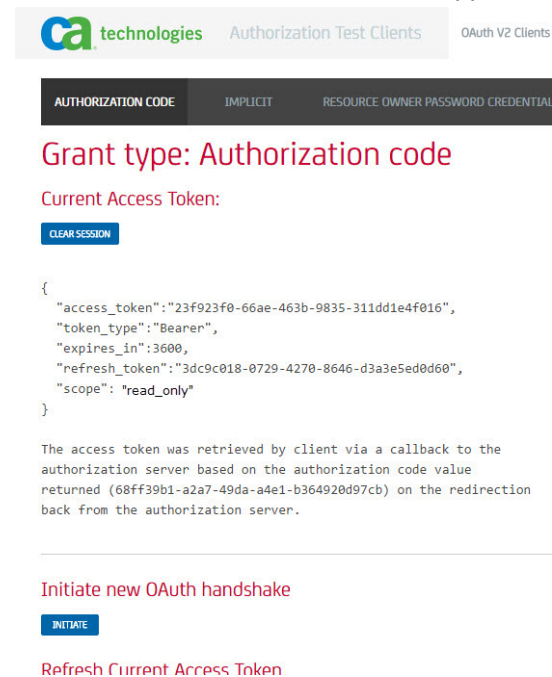
Please grant or deny the request

DENY **GRANT**

© CA Technologies. All rights reserved.

5. Click **Grant**.

You are redirected back to the client application with an access token and a refresh token.



ca technologies Authorization Test Clients OAuth V2 Clients

AUTHORIZATION CODE IMPLICIT RESOURCE OWNER PASSWORD CREDENTIAL

Grant type: Authorization code

Current Access Token:

CLEAR SESSION

```
{
  "access_token": "23f923f0-66ae-463b-9835-311dd1e4f016",
  "token_type": "Bearer",
  "expires_in": 3600,
  "refresh_token": "3dc9c018-0729-4270-8646-d3a3e5ed0d60",
  "scope": "read_only"
}
```

The access token was retrieved by client via a callback to the authorization server based on the authorization code value returned (68ff39b1-a2a7-49da-a4e1-b364920d97cb) on the redirection back from the authorization server.

[Initiate new OAuth handshake](#)

INITIATE

[Refresh Current Access Token](#)

Note that this information is for testing purposes only and should never be displayed in a user-agent.

Use the Access Token to Call an API

To test use the access token to call an API on the CA API Gateway:

1. Click **Call API**. The client app will use the access token currently residing in memory as a credential to call the target API.
2. View the response for this call below the **Target** field.

```
{
  "client_id": "1234-abg-1234-23ce-987548397829x ",
  "subscriber_id": "admin",
  "scope": "read_only",
  "expires_at": 1463009865,
  "accessed_at": 1463006336530,
  "custom": {
    "otk": {"client_type": "confidential", "grant_type": "authorization_code"},
    "portal": {},
    "mag": {},
    "clientkey": {}
  }
}
```

Refresh a Token

Certain grant types support refresh tokens. This is indicated by the presence of a Refresh button.

To refresh an existing OAuth access token, click the **Refresh** button.

The current access token information changes and a note indicates the access token was refreshed.

Clear the Current Session

Click the **Clear Session** button on the OAuth client page. This starts a new test and clears all the parameters in the clients.

Verify the OAuth Infrastructure

The following instructions are for OAuth 2.0 installations only.

Use the OAuth Manager tool to verify the OAuth infrastructure using the OpenID Connect Test Client to request an OAuth 2.0 access_token, refresh_token and an OpenID Connect id_token.

1. Navigate to <https://<your-ssg>:8443/<Prefix>/oauth/v2/client/bcp>
The browser displays a simple OpenID Connect test client using response_type=code.
2. Scroll to the bottom of the page and click **Send**.
The OAuth 2.0 Authorization Server login page appears.
3. Enter admin credentials and click **Login**.
The grant page appears with the client scope permissions. For example: phone email address profile openid.

4. Verify that the scope values do not include "oob".
If "oob" is listed as a scope type, restart the CA API Gateway then restart this verification procedure.
5. Click **Grant**.
Information about the request is displayed.
6. Click either the **Resources** or **Claims** buttons.
A JSON message is displayed as shown below:

access_token: 202c0f69-4bbe-43ac-97e3-fb37664f0c26

Go ahead and use the **access_token**:

Request resources: **RESOURCES** (<https://mag-autotest-mysql.l7tech.com:8443/oauth/v2/protectedapi/foo>)

To access the /userinfo endpoint the additional parameter-value pair 'schema=openid' is required!

Request user claims: **CLAIMS** (<https://mag-autotest-mysql.l7tech.com:8443/openid/connect/v1/userinfo>)

Result after accessing resources or the userinfo endpoint:

```
{
  "sub": "d6qmxEsKY-tc-962lCYWF1f0s-Ko60mPjKTrE9f2vwc",
  "name": "Darth",
  "family_name": "Vader",
  "nickname": "Master-of-the-dark-force",
  "preferred_username": "admin",
  "email": "darth.vader@deathstar.space",
  "phone": "+1-555-123-1234",
  "address": {
    "street": "C-57"
  }
}
```

Userinfo Endpoint Customization

In the OTK Client Context Variables policy, the `path_userinfoendpoint` context variable is set to: `/openid/connect/v1/userinfo`.

Comment: SERVER paths --- visible in the DMZ network =====

- ✔ Set Context Variable `host_oauth2_auth_server` as String to: `https://${gateway.cluster.hostname}:8443`
- ✔ Set Context Variable `path_authorize` as String to: `/auth/oauth/v2/authorize`
- ✔ Set Context Variable `path_token` as String to: `/auth/oauth/v2/token`
- ✔ Set Context Variable `path_userinfoendpoint` as String to: `/openid/connect/v1/userinfo`
- ✔ Set Context Variable `path_resource_endpoint` as String to: `/oauth/v2/protectedapi/resourcefoo`






Comment: SERVER paths --- visible in the Internal network =====

Together, the `host_oauth2_auth_server` and `path_userinfoendpoint` create the full userinfo endpoint: `location_userinfoendpoint`.

To customize the userinfo endpoint:

1. Open the OTK Client Context Variables policy.
2. Copy the Set Context Variable `path_userinfoendpoint`.
3. Open the #OTK Client Context Variables policy. Find this policy in the Customizations folder.
4. Paste the assertion.
5. Double-click the assertion and provide a new endpoint.

If you need to modify the server details specifically for the custom userinfo endpoint, copy the Set Context Variable `host_oauth2_auth_server` and perform the same tasks.

-  Comment: Target Configuration Policy: "OTK Client Context Variable"
-  Comment: === Set custom values for Context Variables below ===
-  Comment: === Set Context Variables to override their values in the target policy below ===
-  Set Context Variable path_userinfoendpoint as String to: /openid/connect/v1/myNewEndpoint
-  Set Context Variable host_oauth2_auth_server as String to: https://myHostname.com:8443

Troubleshooting

This topic provides troubleshooting assistance for common issues. For additional assistance, visit [CA Support](#).

JDBC Connection Failures

My JDBC connection to the database is failing because the server thinks it is under a DDOS attack due to a sudden surge of connections.

Solution: Use the recommended configuration below:

- Ensure that the following JDBC Cluster Properties are set to the same value:
 - jdbcConnection.pooling.maxPoolSize.defaultValue*
 - jdbcConnection.pooling.minPoolSize.defaultValue*
- Set the following Additional Properties for the JDBC connection:
 - Property Name: **maxIdleTime**
Property Value: **0**
To set a C3P0 pooling property: **selected**
 - Property Name: **maxConnectionAge**
Property Value: **0**
To set a C3P0 pooling property: **selected**
 - Property Name: **idleConnectionTestPeriod**
Property Value: **600**
To set a C3P0 pooling property: **selected**
 - Property Name: **EnableCancelTimeout**
Property Value: **true**
To set a C3P0 pooling property: **selected**
- If an Oracle DB is in use, also set this Additional Property:
 - Property Name: **preferredTestQuery**
Property Value: **"select 1 from dual"** (enter the entire value, including the quotation characters)
To set a C3P0 pooling property: **selected**

Deleting OAuth Policy Creates Dependency Errors

When I deleted an OAuth policy, it resulted in many "dependency" errors that needed to be acknowledged one at a time. How can I avoid this?

Solution: This is a result of the Policy Manager's validation checks: it is warning you that you are removing a policy fragment that is currently serving as a backing policy for an encapsulated assertion. To avoid having to acknowledge each dependency error, first delete all the encapsulated assertions related to the OAuth Toolkit.

You can do this by using the Manage Encapsulated Assertion task. See [API Gateway documentation](#). Advanced users can also avoid these errors by deleting the OAuth folder using the RESTMan Service. (SSM-5068).

Upgrade the OTK

These upgrade instructions assume that you have an existing CA API Management OAuth Toolkit (OTK) installation.

To upgrade the OTK:

Before you Begin

WARNING

After you upgrade, restart the Gateway. See [Post-Installation Tasks](#) for more information.

Be aware of the following behavior:

- Verify that the CA API Gateway has been updated to a version that supports the target OAuth Toolkit version. Refer to the Product Compatibility and Support section of the [Release Notes](#).
- The solution kit manager does not check the version of the upgrade. Verify that you are upgrading to a version later than the one currently installed. Unpredictable results occur if you upgrade to an older version.
- Policies with the Customizations folder are not replaced during an upgrade. Your custom configuration remains intact.
- The **Upgrade** button is only functional when upgrading 4.x to later versions. It cannot be used to upgrade versions 3.x. To upgrade versions 3.x, perform an Install. See [Install the OAuth Solution Kit](#).
- To keep the previous version as a reference, **install** the OTK using an instance modifier.
- If you are upgrading OTK solution kits that already have an instance modifier, the same instance modifier is automatically applied to the upgraded version.
- A TimeoutRuntimeException can occur if you do not have enough RAM to perform the upgrade. Minimum memory for OTK should be at least 4 GB. If you have multiple instances of the OTK or other solution kits installed, we suggest minimum of 8 GB for the CA API Gateway RAM.

WARNING

If the TimeoutRuntimeException occurs, OTK might get into a non-removable and non-upgradable state. To restore the system, you will need to clean up OTK and the rest of the solution kits through Gateway Database.

```
Unable to upgrade solution kit: Could not access HTTP invoker remote service...
...TimeoutRuntimeException: Read timed out
```

How are Solution Kits Upgraded?

When you select a product solution kit such as OTKSolutionKit and click **Upgrade**, the list of solution kits within the product solution kit appear for selection.

When you select solution kits for upgrade, expect the following behavior:

- If the selected solution kit was previously installed, the solution kit is **upgraded**.
- If the selected solution kit was not previously installed, the solution kits is **installed**.
- Previously installed solution kits that are not selected for Upgrade are **uninstalled**.
For example, if you had the Shared Portal Resources solution kit installed in OTK 4.2 and do not select Shared Portal Resources during the upgrade to OTK 4.3, the Shared Portal Resources solution kit is uninstalled.

How Are Services Upgraded?

Services are replaced with new versions during an upgrade. However, your **service revision history** remains intact, allowing you to compare versions and make edits.

WARNING

If you are an API Management SaaS customer, any changes you made to OTK Services endpoints are overwritten after the automatic upgrade. OTK will be upgraded automatically when a new release is available.

To view the history of a service and compare revisions:

1. Select the service.
 2. Right-click and select **Revision History**.
 3. In the Policy Revisions dialog, select the active version. The Active version of the service is indicated with *.
 4. Click the **Compare Policy** button.
 5. Select the previous version of the service.
 6. Click the **Compare Policy** button.
- A window appears showing you the modifications.

NOTE

No history is maintained for OAuth 1.0 services. OAuth 1.0 is deprecated. Services are replaced by an upgrade.

Upgrade from OTK Version 4.x

You can upgrade from OTK version 4.x directly to any later version. For example, you can upgrade from OTK 4.1 to OTK 4.3, without having to upgrade to OTK 4.2. If you are upgrading from OTK version 4.0, you must first run a compatibility patch. This procedure is described in the instructions.

To upgrade from OTK version 4.x to a later version, perform the upgrade tasks in the following order:

Upgrade the OTK Database

To upgrade an existing OTK database to a later version:

1. Download the database update scripts found on [Supporting Files](#).
To upgrade an existing database, start with the script corresponding to your current OTK version, then work up the list, executing all scripts until you reach the latest version.
2. Follow the upgrade instructions for your database type.
See [Create or Upgrade the OTK Database](#)

Run the Upgrade Compatibility Patch (OTK 4.0.x only)

The upgrade compatibility patch is required only if you are upgrading from OTK 4.0.x. This patch is not required when upgrading from later versions of the OTK. The patch is available on [Supporting Files](#).

To run the upgrade compatibility patch:

1. Open a privileged shell on the CA API Gateway. The upgrade compatibility patch is run against the CA API Gateway database.
2. Copy the `otk_upgrade_compatibility_4.0.sh` script to the CA API Gateway.
3. On the Gateway, run the following command:

```
./otk_upgrade_compatibility_4.0.sh
```
4. Answer the following prompts by providing CA API Gateway database details:
 - a. Enter the database hostname
 - b. Enter the database port
 - c. Enter the database name
 - d. Enter the database user

- e. Enter the database password
5. Restart the Gateway using the following command:

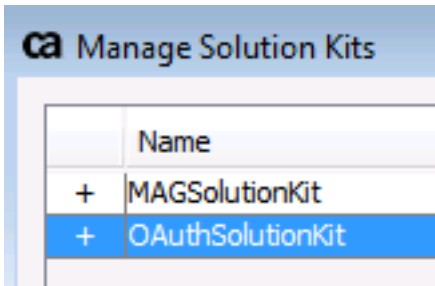

```
service ssg restart
```

Upgrade the OAuth Solution Kits

Before upgrading, see [How are Solution Kits Upgraded?](#)

To upgrade the OAuth Solution Kit:

1. In the Policy Manager, go to **Tasks, Extensions and Add-Ons, Manage Solution Kits**. The list of installed solution kits appears.
2. Click to highlight the existing **OAuthSolutionKit**.



3. Click **Upgrade**.
4. Select the new OAuthSolutionKit.sskar file.
5. Click **Next**. The solution kits within the sskar file appear.
6. Select the solution kits to upgrade or install. For details on how the upgrade affects existing solution kits, see the Solution Kit Upgrade Behavior note at the beginning of this section
7. Click **Next**.
8. Resolve any entity conflicts and click **Finish**.

Upgrade from OTK version 3.x

The **Upgrade** button in the solution kit manager is not functional for these versions. You must perform an **Install**.

To install the latest version and keep the older version as a reference for merging custom configuration:

1. Run the database upgrade scripts.
2. **Install** version 4.x through the solution kit manager using an instance modifier.
3. Merge any custom configuration into the new #policies and extensions found in the Customizations folder.

To install the latest version without keeping the older version:

1. Run the database upgrade scripts.
2. Manually **Uninstall** OTK 3.x, remove any remaining empty folders. See the "Uninstall Installations Prior to OTK Version 3.2" section of [Uninstall the OTK](#).
3. After the old version is removed, including any empty folders, **install** the latest OTK 4.x version. Configuration must be manually performed in the OTK 4.x version.

Upgrade with RESTman instead of the Solution Kit Installer

The Solution Kit Manager accessed through the Policy Manager provides a convenient interface for installing and removing solution kits.

Alternatively, if you have the REST Management Service (RESTman) running on the CA API Gateway, you can access the REST Management API and run a script to upgrade the OTK. For more information, see the [REST Management API](#) documentation on the CA API Gateway site.

Download the upgrade scripts from [Download OTK Installation Files](#).

NOTE

Currently, the only upgrade script available is: `upgrade-to-otk4.3.1.sh`

The script will throw errors if the following requirements are not met:

- Restman must be installed.
- mySQL and curl must be installed
- The following files must be in the same directory as the `upgrade-to-otkversion.sh` script:
 - OTK database upgrade script
 - OTK sskar file

To upgrade OTK using the REST Management Service, run the upgrade script with the required variables. Optional variables are also available.

The following example uses the `upgrade-to-otk4.3.1.sh` script:

```
./upgrade-to-otk4.3.1.sh -l <gateway_with_port> -u <restman_user> -p <restman_password>
-o <otk_db_host> -a <otk_db_password>
```

Required Variables	Notes
<code>-l <gateway_with_port></code>	The host address of the Gateway. For example: myGateway.com:8443
<code>-u <restman_user></code>	The username to access RESTman
<code>-p <restman_password></code>	The password to access RESTman
<code>-o <otk_db_host></code>	The name of the OTK database. Default value: otk_db
<code>-a <otk_db_password></code>	The OTK database password.

The following additional variables for the script are optional.

Optional Variables	Notes
<code>-j <oauth_jdbc_connection_name></code>	The OAuth JDBC connection name to the database. Default value: OAuth.
<code>-s <path_to_oauth_skar></code>	The absolute path to the sskar file. Default value: OAuthSolutionKit.sskar.
<code>-e <otk_db_user></code>	The database user. Default value: otk_user
<code>-n <otk_db_name></code>	The database name. Default value: otk_db
<code>-r <otk_db_port></code>	The database port. Default value: 3306

Perform Post-Installation Tasks

[Post-installation tasks](#) include restarting the Gateway to make sure your upgrade modifications take effect.

Uninstall the OTK

The uninstall procedure performed through the solution kit manager removes installed solution kits and their associated policies.

Clicking the Uninstall button **does not** remove the following:

- Policies inside the Customizations folder
- Customizable policies in the Policy Fragments folder
- OAuth 2.0 Services
- The OTK `v2GenerateRequestMac` policy
- The `V2GenerateAuthHeaderForClient` policy

OAuth 1.0 Services are removed.

The policies inside the Customizations folder include any custom configuration. To completely uninstall the OTK, manually delete the customizations folder, Policy Fragments folder, and Server folder.

To uninstall the OTK, select one of the following tasks:

Uninstall Previously Installed Solution Kits

The Solution Kit manager allows you to uninstall the entire OTK Solution kit, or individual solution kits within the parent solution kit.

To uninstall solution kits:

1. Go to **Tasks, Extensions and Add-Ons, Manage Solution Kits**.
2. Click to select the component you want to uninstall. If you want to remove the entire OTK, select the top level OTK Solution Kit.
3. Click **Uninstall** and confirm.
4. Click **OK** when the task is completed.

The uninstall process leaves empty folders in the policy manager.

To remove the empty folders:

1. In Policy Manager, locate the root folder that you want to uninstall.
2. Right-click the folder in policy manager and select **Delete Folder**.
3. Click the check box to activate the OK button, then click **OK**.
Any empty sub-folders are also deleted.

Uninstall Non Solution Kit Installations of the OTK

OTK versions prior to OTK version 3.2 were not installed using the solution kit installer. Subsequently, uninstalling the OAuth Toolkit is a manual task.

To uninstall OTK versions that were not installed as solution kits :

1. Login into Policy Manager as an administrator.
2. Select the main OAuth folder.
3. Right-click and select **Delete Folder**.
4. Click the check box to activate the OK button, then click **OK**.

If the folder is not deleted, delete components within the folder first, then delete the empty OTK folders.

Prepare JSON Message for Export

Exporting the current OAuth configuration in JSON format provides convenient policy-driven setting of attributes for communication between the CA API Gateway and the OAuth Client. The exported JSON message provides a snapshot of the values assigned to context variables. See [JSON Message Example](#). It also includes endpoint paths and certificate information.

Configuration is exported for a specific client id.

Include the OAuth Server Certificate in the JSON Message

The OAuth server certificate is the SSL certificate on the authorization server. The SSL certificate name must match the `oauth2_server_certificate` variable value in the policy used to create the JSON message.

By default, the certificate name is set to the localhost: `#{gateway.cluster.hostname}`. If your SSL certificate has an alternate name, provide custom configuration in the #OTK Variable Configuration policy.

To include the OAuth Server Certificate:

1. Determine your SSL certificate name. Go to **Tasks, Certificates, Keys and Secrets, Manage Certificates**. Copy the Certificate Name value of the SSL certificate.
2. Is your certificate name the same as your localhost? If so, no additional customization is required. Otherwise, continue to step 3.
3. Open the OTK Variable Configuration policy.
4. Copy the Set Context Variable `oauth2_server_certificate` assertion.
5. Open the #OTK Variable Configuration policy and paste the assertion.
6. Double-click the assertion and assign the certificate name value to the `oauth2_server_certificate` variable.

The name used in the JSON content variable must match the alias of the certificate.

Assign Custom OAuth Server Configuration

To assign OAuth server configuration:

1. Open the OTK Variable Configuration policy.
2. Locate the Export section.

Copy any of the following Set Context Variable assertions that you want to modify. Paste them into the #OTK Variable Configuration policy, and provide custom values.

Context Variable	Notes
<code>oauth2_server_hostname</code>	Hostname of the OAuth server Default setting: <code>#{gateway.cluster.hostname}</code>
<code>oauth2_server_certificate</code>	OAuth server certificate name This name must match the assigned Certificate Name value found in Tasks, Certificates, Keys and Secrets, Manage Certificates . Default setting: <code>#{gateway.cluster.hostname}</code>
<code>oauth2_server_port</code>	The port of the OAuth server where OAuth clients connect Default setting: 8443

Context Variable	Notes
oauth2_server_url_prefix	Prefix of the OAuth server. The instance modifier if used.
expose_client_secret	Boolean Default: false If true, the client secret is included in the exported JSON message
enable_anonymous_client_export	Boolean Default: false. If true, the endpoint allows access to users without authentication. See Allow Anonymous Access to the JSON Export Endpoint .
export_custom	A variable used to pass any further information. By default, this variable passes the following endpoint paths: "oauth_demo_protected_api_endpoint_path": "/oauth/v2/protectedapi/foo" "mag_demo_products_endpoint_path": "/protected/resource/products"

Enable the Export Endpoint

The endpoint that provides configuration values in JSON format is located in the OAuth/DMZ/OAuth 2.0 folder:

/auth/oauth/v2/client/export

By default this endpoint is disabled. The preferred method of exporting the JSON file is through the Export button in OAuth Manager. You do not need to enable this endpoint to export the JSON file through OAuth Manager.

To enable the export endpoint:

1. Locate the /auth/oauth/v2/client/export endpoint. By default, this is located in Server/DMZ/OAuth 2.0/TokenServer
2. Right-click the endpoint, and select **Service Properties**.
3. Select **Enable** and click **OK**.

Allow Anonymous Access to the JSON Export Endpoint

By default the JSON export endpoint allows access to users authenticated using HTTP Basic Authentication only.

To allow anonymous access to the JSON export endpoint:

1. Open the **OTK Variable Configuration** policy.
2. Copy the Set Context Variable enable_anonymous_client_export Context Variable.
3. Open the **#OTK Variable Configuration** policy and paste the assertion.
4. Modify the value to **true**.

Both authorized and anonymous users require a secure SSL connection to access the endpoint.

JSON Message Example

The JSON message contains the current server configuration values used to initialize the SDK for a client. The message comprises four sections:

Server

The server section contains server details and the SSL certificate required to establish communication between the SDK and the MAG. It also contains the OAuth server SSL certificate information in the `server_certs` attribute. The name of the certificate is identified in the OTK Variable Configuration policy. See [Prepare JSON Message for Export](#).

```
"server": {
  "hostname": "example.com",
  "port": 8443,
  "prefix": "myPrefix",
  "server_certs": [
    [
      "-----BEGIN CERTIFICATE-----",
      "MIIC9TCCA...", "bi5...", "aW4....",
      "-----END CERTIFICATE-----"
    ]
  ]
}
```

MAG

MAG and the OAuth Manager Extension must be installed for the Mobile API Gateway section to be included in the JSON configuration file.

```
"mag": { "system_endpoints":
  { "device_register_endpoint_path": "/connect/device/register",
    "device_remove_endpoint_path": "/connect/device/remove",
    "client_credential_init_endpoint_path": "/connect/client/initialize" },
  "oauth_protected_endpoints":
  { "enterprise_browser_endpoint_path": "/connect/enterprise/browser",
    "device_list_endpoint_path": "/connect/device/list" },
  "mobile_sdk":
  { "sso_enabled": true,
    "location_enabled": true,
    "location_provider": "network",
    "msisdn_enabled": true,
    "trusted_public_pki": true,
    "trusted_cert_pinned_public_key_hashes": [],
    "client_cert_rsa_keybits" : 1024
  }
}
```

No explicit configuration of server variables for export is required. However, you can override existing values by editing the MAG Variable Configuration policy found in /Policy Fragments/configuration.

Exported Attributes	Description	Type
device_register_endpoint_path	URL suffix for device registration	String
device_remove_endpoint_path	URL suffix for token server's remove_device_x509 endpoint	String

Exported Attributes	Description	Type
client_credential_init_endpoint_path	URL Suffix for initialize client credential endpoint	String
enterprise_browser_endpoint_path	URL suffix for server's enterprise apps endpoint.	String
device_list_endpoint_path	URL suffix for device list	String
sso_enabled	Indicates whether single sign on is allowed for this app.	Boolean
location_enabled	Indicates whether location information is allowed in outbound requests	Boolean
location_provider	The location provider to use if location is enabled	String
msisdn_enabled	MSISDN information should be included in the outbound requests	Boolean
trusted_public_pki	Indicates whether public CAs recognized by the OS are accepted as TLS server certificates in addition to the list returned by server_certs	Boolean
trusted_cert_pinned_public_key_hashes	Controls whether TLS server certificate public key pinning is in use and, if so, what pinned public key hashes to permit within server cert chains	JSON Array of JSON Array of String
client_cert_rsa_keybits	The size in bits of the RSA keypair to generate for the client certificate	Number

OAuth

For the OAuth section to be included in the JSON configuration file, OAuth Manager must be installed.

```
"oauth": {
  "client": {
    "organization": "CA Technologies",
    "description": "Example application for Mobile SSO demonstrations",
    "client_name": "AppA",
    "client_type": "confidential",
    "registered_by": "admin",
    "client_ids": [
      {
        "client_id": "12341234b4f-aaaa-aaab-aaab-123412345377a",
        "client_secret": "abababa25-1239-1232-1232-12345678999a",
        "scope": "openid msso phone profile address email msso_register",
        "redirect_uri": "https://android.ssosdk.ca.com/android",
        "environment": "Android",
        "status": "ENABLED",
        "registered_by": "admin"
        "service_ids": "",
        "account_plan_mapping_ids": "",
        "client_key_custom": "{}"
      }
    ]
  }
}
```

```

    }
  ]
},
"system_endpoints": {
  "authorization_endpoint_path": "/auth/oauth/v2/authorize",
  "token_endpoint_path": "/auth/oauth/v2/token",
  "token_revocation_endpoint_path": "/auth/oauth/v2/token/revoke",
  "usersession_logout_endpoint_path": "/connect/session/logout"
  "usersession_status_endpoint_path": "/connect/session/status"
},
"oauth_protected_endpoints": {
  "userinfo_endpoint_path": "/openid/connect/v1/userinfo",
  "usersession_status_endpoint_path": "/connect/session/status"
}
}

```

You are required to configure certain variables in the #OTK Variable Configuration policy. See [Prepare JSON Message for Export](#).

Open the #OTK Variable Configuration policy found in the OAuth/Policy Fragments/configuration/ to further customize values for the JSON message.

Exported Attributes	Description	Type
organization	The organization name to include in the client cert DN	String
description	App description	String
name	Client name	String
type	Client type	String
registered_by	User who registered the client	String
client_ids	List of client ids, only the first client_id will be used by the SDK and the rest will be ignored by the SDK	JSON Array
client_id	The application's client id for the initial OAuth token request	String
client_secret	The application's client secret for the initial OAuth token request	String
scope	The OAuth scope string that should be requested when obtaining an access token that will be used to consume service from an API endpoint	String
redirect_uri	The redirect URI provided to the third-party-login platform.	String
environment	The environment of the client	String
status	Status of the client	String
registered_by	User who registered the client	String
authorization_endpoint_path	System endpoint for authorization	String
token_endpoint_path	System endpoint to acquire token	String
token_revocation_endpoint_path	System endpoint to revoke token	String

Exported Attributes	Description	Type
usersession_logout_endpoint_path	System endpoint to perform logout user session	String
userinfo_endpoint_path	Endpoint to retrieve user info	String
usersession_status_endpoint_path	Endpoint to check user session status	String

Custom

The custom section allows you add additional attributes to pass values to the SDK.

```
"custom": {  
  "oauth_demo_protected_api_endpoint_path": "/oauth/v2/protectedapi/foo",  
  "mag_demo_products_endpoint_path": "/protected/resource/products"  
}
```

To add new attributes to the Custom section of the JSON content, set the `export_custom` variable value in the #OAuth Variable Configuration policy.

Secure an API Endpoint with OAuth 2.0

Use the OTK Require OAuth 2.0 Token encapsulated assertion to secure an API endpoint.

NOTE

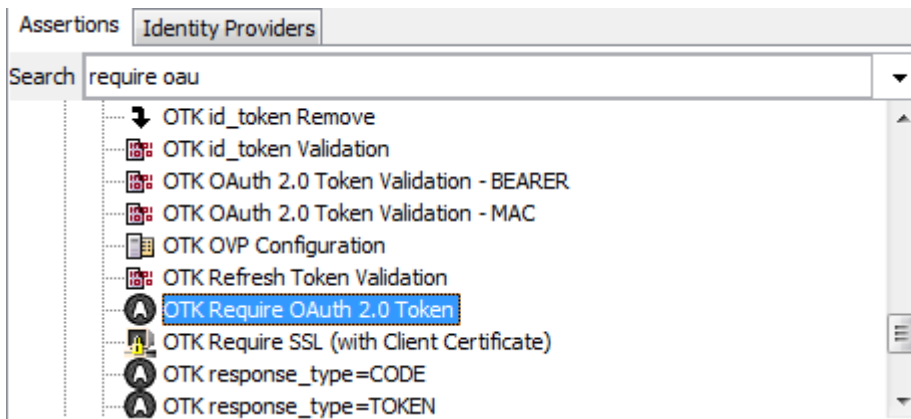
OAuth 1.0 is no longer supported.

OTK Require OAuth 2.0 Token

Use the OTK Require OAuth 2.0 Token encapsulated assertion to allow access to an API only when a valid OAuth 2.0 access_token is presented by the client.

This assertion includes the OTK Access Token Retrieval assertion to locate the incoming OAuth 2.0 access_token.

Place the assertion as early as possible in an API policy.



Drag the assertion into a policy and configure the properties shown in the table below.

Properties	Parameter Name	Type	Notes
Required SCOPE(s)	scope_required	String	If SCOPE is not required, this value can be empty. A space separated list of required SCOPES. An access_token is only accepted if it has been granted with those SCOPE values.

Cache validation result (s)	cache_lifetime	Integer	This value cannot be empty. Represents the time in seconds for which an access_token is cached. The assertion initially validates an access_token. The validation result is then cached until the cache period expires. This increases performance, but also enables clients to use potentially expired access_tokens. The cache_lifetime value extends the lifetime of the token. A value of 0 indicates no caching is performed.
Is this a one-time access-token?	onetime	Boolean	Default value: false . To allow an access_token to be considered valid only once for this endpoint, set this value to true . This setting is rare, but enables special use cases.
Fail if this SCOPE was granted?	scope_fail	Boolean	Default value: false . Set this value to "true" if a request should fail in the case that an access_token has been granted for at least one the specified SCOPE values listed above
Access Token	given_access_token	String	Optional. The hardcoded value of an access token or a context variable representing an access_token. Use this property if an access_token is made available, but not by the client, or if the access_token is passed using a non-standard mechanism.

OAuth Request Scenarios

The following scenarios examine OAuth 2.0 client requests for an `access_token`.

Response Type

The client request includes a `response_type`.

response_type	Notes
token	Only use the token response_type if an exposed <code>access_token</code> is not an issue. Clients using this response_type are considered to be public clients and do not receive a <code>refresh_token</code> . This type of client may be implemented in JavaScript.
token id_token	Only use the token id_token response type in conjunction with <code>scope=openid</code> . This response_type is used with OpenID Connect.
code	The code response_type is the most secure with regards to visibility of issued tokens. The flow involves multiple steps that are required between sending the initial request to receiving an <code>access_token</code> . If the user grants the client access, the client receives an authorization code attached to the <code>redirect_uri</code> . The client can then exchange the <code>authorization_code</code> for an <code>access_token</code> using <code>grant_type=authorization_code</code> .

Details:

response_type=token

The 'token' response_type can be used when the client should not have access to user credentials. It includes a redirect that involves a browser or a web view on a mobile device. Using the token response_type is not secure. It should be used only if an exposed `access_token` is not an issue. Clients using this response_type are considered to be 'public' clients and do not receive a `refresh_token`. This type of client may be implemented in JavaScript.

Request	Notes
Method:	GET
Endpoint:	/auth/oauth/v2/authorize
Parameters:	<code>response_type=token&client_id=a-client_id&redirect_uri=a-redirect_uri&scope=a-list-of-scope-values&state=a-state-value</code>
Optional:	redirect_uri: If provided only requests using a registered <code>redirect_uri</code> of this client will be granted by the OAuth server. If the parameter is not included the OAuth server will use the registered <code>redirect_uri</code> . If multiple <code>redirect_uris</code> have been registered the request will fail. At least one <code>redirect_uri</code> MUST have been registered!
Optional:	scope: Only SCOPE values that have been registered for the client will be granted by the OAuth server
Optional:	state: This value is opaque to the OAuth server and will be passed back unmodified to the client

Response	Notes
Header:	status: 200
Header:	content-type: text/html
Body:	The user-agent will receive a login page. This page will request user credentials and the consent of the user. If the user denies the request the client will receive an error. If the user grants the client it will receive the access_token attached to the redirect_uri
Next:	The OAuth server will redirect the user-agent back to the client:
Header:	302
Header:	Location: the-redirect-uri?state=the-given-state#access_token=an-access_token&expires_in=lifetime-in-seconds&token_type=Bearer&scope=granted-scope

The receiving user-agent (browser, JavaScript client) can now extract the parameters from the redirect_uri fragment portion. The fragment value will only be available in the browser.

response_type=token id_token

The 'token id_token' response_type can be used when the client should not have access to user credentials. It includes a redirect that involves a browser or a web view on a mobile device. Using the token response_type is not secure. It should be used only if an exposed access_token is not an issue. Clients using this response_type are considered to be 'public' clients and do not receive a refresh_token. This type of client may be implemented in JavaScript.

Request	Notes
Method:	GET
Endpoint:	/auth/oauth/v2/authorize
Parameters:	response_type=token%20id_token&client_id=a-client_id&redirect_uri=a-redirect_uri&state=a-state-value&scope=a-list-of-scope-values (SCOPE MUST be included and it MUST include 'openid')
Optional:	redirect_uri: If provided only requests using a registered redirect_uri of this client will be granted by the OAuth server. If the parameter is not included the OAuth server will use the registered redirect_uri. If multiple redirect_uris have been registered the request will fail. At least one redirect_uri MUST have been registered!
Optional:	state: This value is opaque to the OAuth server and will be passed back unmodified to the client

Response	Notes
Header:	status: 200
Header:	content-type: text/html
Body:	The user-agent will receive a login page. This page will request user credentials and the consent of the user. If the user denies the request the client will receive an error. If the user grants the client it will receive the access_token attached to the redirect_uri

Response	Notes
Next:	The OAuth server will redirect the user-agent back to the client:
Header:	302
Header:	Location: the-redirect-uri?state=the-given-state#access_token=an-access_token&expires_in=lifetime-in-seconds&token_type=Bearer&scope=granted-scope&id_token=an-id-token-represented-as-jwt&id_token_type=urn%3Aietf%3Aparams%3Aoauth%3Agrant-type%3Ajwt-bearer

The receiving user-agent (browser, JavaScript client) can now extract the parameters from the `redirect_uri` fragment portion. The fragment value will only be available in the browser.

response_type=code

This is the safer `response_type` to use because it is the most secure with regards to visibility of issued tokens. The flow involves multiple steps that are required between sending the initial request to receiving an `access_token`

A client is requesting an `access_token` using `response_type=code`. This `response_type` can be used if the client should not have access to user credentials. This `response_type` includes a redirect that involves a browser or a web view on a mobile device.

Request	Notes
Method:	GET
Endpoint:	/auth/oauth/v2/authorize
Parameters:	<code>response_type=code&client_id=a-client_id&redirect_uri=a-redirect_uri&scope=a-list-of-scope-values&state=a-state-value</code>
Optional:	redirect_uri: If provided only requests using a registered <code>redirect_uri</code> of this client will be granted by the OAuth server. If the parameter is not included the OAuth server will use the registered <code>redirect_uri</code> . If multiple <code>redirect_uris</code> have been registered the request will fail. If a <code>redirect_uri</code> is included and none was registered the OAuth server will use the one included in the request
Optional:	scope: Only SCOPE values that have been registered for the client will be granted by the OAuth server
Optional:	state: This value is opaque to the OAuth server and will be passed back unmodified to the client

Response	Notes
Header:	status: 200
Header:	content-type: text/html
Body:	The user-agent will receive a login page. This page will request user credentials and the consent of the user. If the user denies the request the client will receive an error. If the user grants the client it will receive an authorization code attached to the <code>redirect_uri</code>
Next:	The OAuth server will redirect the user-agent back to the client:
Header:	302

Response	Notes
Header:	Location: the-redirect-uri?code=an-authorization-code&state=the-given-state

The receiving client can now extract the code (`authorization_code`) from the `redirect_uri` and exchange it for an `access_token` (using `grant_type=authorization_code`).

Grant Types

The following scenarios describe the different grant types used to request an `access_token`.

grant_type	Notes
password	Used if the client was built by the enterprise that also implements the OAuth token server.
authorization_code	Exchange the <code>authorization_code</code> for an <code>access_token</code> . A client has received the <code>authorization_code</code> attached to a redirect URI. The client now exchanges the <code>authorization_code</code> for an <code>access_token</code> by using <code>grant_type 'authorization_code'</code> . If the client included 'openid' as SCOPE in his request, additional keys are included in the response: ... <code>"id_token": "eyJ0eXAiOiV8 ... JZu_LsN851VtFC5pqlqJc",</code> <code>"id_token_type": "urn:ietf:params:oauth:grant-type:jwt-bearer" ...</code> The <code>id_token</code> (JWT) can be used with <code>grant_type=urn:ietf:params:oauth:grant-type:jwt-bearer</code> .
urn:ietf:params:oauth:grant-type:jwt-bearer	Used if the client has an <code>id_token</code> (represented as JWT) of an authenticated user. Only <code>id_tokens</code> issued by the OAuth server are accepted.
urn:ietf:params:oauth:grant-type:saml2-bearer	This <code>grant_type</code> can be used if the client is in possession of a SAML 2.0 token of an authenticated user. This scenario is useful in cases of federation where the SAML 2.0 token was signed by a trusted party.
refresh_token	This <code>grant_type</code> can be used if the client is in possession of a <code>refresh_token</code> . The request will only be successful if the <code>refresh_token</code> has not expired. The parameter 'SCOPE' can only include the same or a subset of values that were originally requested. By default a <code>refresh_token</code> can be used only once.

Details:

grant_type=password

This `grant_type` can be used if the client was built by the enterprise that also implements the OAuth token server.

Request	Notes
Method:	POST
Header:	content-type: application/x-www-form-urlencoded
Header:	authorization: Basic base64(client_id:client_secret) (This header can only be used if 'client_id' and 'client_secret' are NOT found within the message body and vice versa!)
Endpoint:	/auth/oauth/v2/token

Request	Notes
Parameters:	grant_type=password&username=a-username&password=a-user-s-password&client_id=a-client_id&client_secret=a-client_secret&scope=a-list-of-scope-values
Optional:	scope: Only SCOPE values that have been registered for the client will be granted by the OAuth server

Response	Notes
Header:	status: 200
Header:	content-type: application/json
Body:	Example: { "access_token":"115b8c ... 11a5", "token_type":"Bearer", "expires_in":3600, "refresh_token":"74b29d19-8b ... 7bb6bd1", "scope":"openid email" }

grant_type=client_credentials

This grant_type can be used if the client is acting on its own behalf. No user consent is required.

Request	Notes
Method:	POST
Header:	content-type: application/x-www-form-urlencoded
Header:	authorization: Basic base64(client_id:client_secret) (This header can only be used if 'client_id' and 'client_secret' are NOT found within the message body and vice versa!)
Endpoint:	/auth/oauth/v2/token
Parameters:	Parameters: grant_type=client_credentials&client_id=a-client_id&client_secret=a-client_secret&scope=a-list-of-scope-values
Optional:	scope: Only SCOPE values that have been registered for the client will be granted by the OAuth server

Response	Notes
Header:	status: 200
Header:	content-type: application/json
Body:	{ "access_token":"115b8c ... 11a5", "token_type":"Bearer", "expires_in":3600, "scope":"openid email" }

grant_type=authorization code

Exchange the authorization_code for an access_token. A client has received the authorization_code attached to a redirect URI. The client now exchanges the authorization_code for an access_token by using grant_type 'authorization_code'.

Request	Notes
Method:	POST
Header:	content-type: application/x-www-form-urlencoded
Header:	authorization: Basic base64(client_id:client_secret) (This header can only be used if 'client_id' and 'client_secret' are NOT found within the message body and vice versa!)
Endpoint:	/auth/oauth/v2/token
Parameters:	grant_type=authorization_code&code=the-received-authorization-code&client_id=a-client_id&client_secret=a-client_secret&redirect_uri
Optional:	redirect_uri: The value has to be included if it has been used in the initial request. It also has to match the original value

Response	Notes
Header:	status: 200
Header:	content-type: application/json
Body:	{ "access_token":"115b8c ... 11a5", "token_type":"Bearer", "expires_in":3600, "refresh_token":"74b29d19-8b ... 7bb6bd1", "scope":"openid email" }

If the client included 'openid' as SCOPE in his request, additional keys are included in the response:

... "id_token":"eyJ0eXAiOiVv8 ... JZu_LsN851VtfC5pqlqJc", "id_token_type":"urn:ietf:params:oauth:grant-type:jwt-bearer" ...

The id_token (JWT) can be used with grant_type=urn:ietf:params:oauth:grant-type:jwt-bearer.

grant_type=urn:ietf:params:oauth:grant-type:jwt-bearer

This grant_type can be used if the client is in possession of an id_token (represented as JWT) of an authenticated user. Only id_token (JWT) that were issued by the OAuth server are accepted.

Request	Notes
Method:	POST
Header:	content-type: application/x-www-form-urlencoded
Header:	authorization: Basic base64(client_id:client_secret) (This header can only be used if 'client_id' and 'client_secret' are NOT found within the message body and vice versa!)
Endpoint:	/auth/oauth/v2/token
Parameters:	grant_type=urn%3Aietf%3Aparams%3Aoauth%3Agrant-type%3Ajwt-bearer&assertion=a-jwt&client_id=a-client_id&client_secret=a-client_secret&scope=a-list-of-scope-values
Optional:	scope: Only SCOPE values that have been registered for the client will be granted by the OAuth server

Response	Notes
Header:	status: 200
Header:	content-type: application/json
Body:	{ "access_token":"115b8c ... 11a5", "token_type":"Bearer", "expires_in":3600, "refresh_token":"74b29d19-8b ... 7bb6bd1", "scope":"openid email" }

grant_type=urn:ietf:params:oauth:grant-type:saml2-bearer

This grant_type can be used if the client is in possession of a SAML 2.0 token of an authenticated user. This scenario is useful in cases of federation where the SAML 2.0 token was signed by a trusted party.

Request	Notes
Method:	POST
Header:	content-type: application/x-www-form-urlencoded
Header:	authorization: Basic base64(client_id:client_secret) (This header can only be used if 'client_id' and 'client_secret' are NOT found within the message body and vice versa!)
Endpoint:	/auth/oauth/v2/token
Parameters:	grant_type=urn%3Aietf%3Aparams%3Aoauth%3Agrant-type%3Asaml2-bearer&assertion=a-base64-encoded-saml-token&client_id=a-client_id&client_secret=a-client_secret&scope=a-list-of-scope-values
Optional:	scope: Only SCOPE values that have been registered for the client will be granted by the OAuth server

Response	Notes
Header:	status: 200
Header:	content-type: application/json
Body:	{ "access_token":"115b8c ... 11a5", "token_type":"Bearer", "expires_in":3600, "scope":"openid email" }

grant_type=refresh_token

This grant_type can be used if the client is in possession of a refresh_token. The request will only be successful if the refresh_token has not expired. The parameter 'SCOPE' can only include the same or a subset of values that were originally requested. The refresh_token can only be use once.

Request	Notes
Method:	POST
Header:	content-type: application/x-www-form-urlencoded
Header:	authorization: Basic base64(client_id:client_secret) (This header can only be used if 'client_id' and 'client_secret' are NOT found within the message body and vice versa!)
Endpoint:	/auth/oauth/v2/token

Request	Notes
Parameters:	Parameters: grant_type=refresh_token&refresh_token=a-refresh-token&client_id=a-client_id&client_secret=a-client_secret&scope=a-list-of-scope-values
Optional:	scope: Only SCOPE values that have been registered for the client will be granted by the OAuth server

Response	Notes
Header:	status: 200
Header:	content-type: application/json
Body:	{ "access_token":"115b8c ... 11a5", "token_type":"Bearer", "expires_in":3600, "refresh_token":"74b29d19-8b ... 7bb6bd1", "scope":"openid email" }

Customizing the OAuth Toolkit

All OAuth Toolkit features are implemented using policies in the Policy Manager, making it possible to customize the entire OAuth process.

Customizing Policies

All OAuth Toolkit features are implemented using policies in the Policy Manager, making it possible to customize the entire OAuth process.

Understanding Customization

The CA OAuth Toolkit is installed with default settings. Customization is your configuration of these settings. Some customization is required while other customization is optional. Starting in version 4.0, all customization is performed in the #policies and the extensions located in the Customizations folder.

#Policies

Each editable #policy in the Customizations folder has a target read-only policy in the Policy Fragments folder.

For example, the **#OTK Variable Configuration** policy contains customizations for the **OTK Variable Configuration** target policy. Customizations are included by reference in the target policy.

NOTE

Values set in the #policy override default values in the corresponding target policy.

Extensions

Extensions handle more complex configuration. They contain policy logic overrides and are not prefixed with the # character. They may contain assertions to double-click, disabled code to enable, and require flags to be set in the #policy. Open the extension and read the comments for instructions specific to each extension.

For example, the **OTK User Authentication Extension** contains both logic and disabled assertions. The extension targets the read-only **OTK User Authentication** policy.

Some extensions, such as the **OTK CORS extension** contain no code. Configure the extension as described by the documentation within this site.

How to Migrate Your Existing Policy Customizations

In previous versions, you configured OTK by editing the default installed policies directly. Custom values for required and optional configuration became part of the installed policies. Performing an upgrade required overwriting your existing policies with new policies containing default configuration. Any customization was lost, unless you installed the latest version with an instance modifier, compared the old and new policies, and manually copied over your customizations.

Starting in version 4.0, your customizations are stored safely outside of the installed default policies and are not overwritten by upgrades.

To migrate your existing pre-4.0 policy configuration into the 4.0 workflow, consider the following strategies:

- Read through your existing policies and record any custom settings. Install version 4.0 and transfer your custom settings to the #policies and extensions in the new Customizations folder. The documentation on this site provides instructions on how to configure functionality within the Customizations folder.
- Consider installing version 4.0 using an index modifier. Your new 4.0 policies will differ considerably from your old. Locate any Context Variables with custom values. For each policy, set these values in the corresponding #policy.

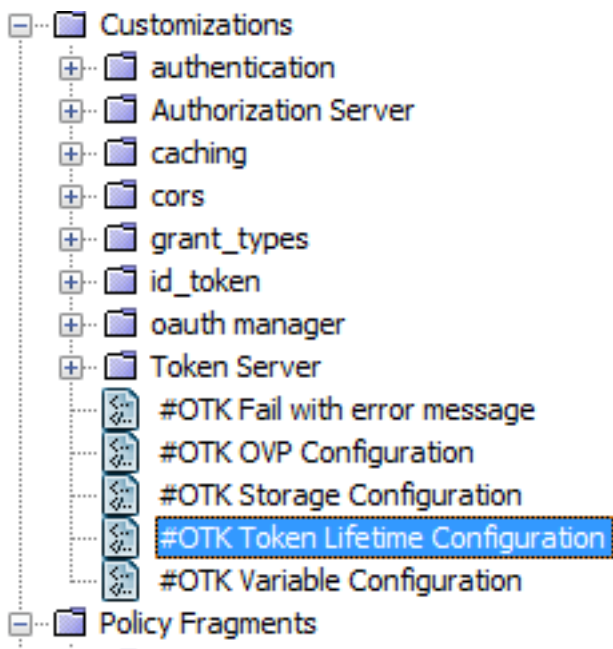
Upgrading After 4.0

After migrating your customizations to the 4.0 framework, upgrade is easier. When policies in the Policy Fragments folder are replaced to support new features, your customizations remain intact. Your custom values still override default values in the newly upgraded policies. You no longer have to re-enter your hostname and database type, for example.

New features inevitably come with new context variables set to default values. Certain context variables may be deprecated. New logic may be introduced. You must maintain your #policies and extensions to accommodate these changes and provide custom configuration. As always, instructions for configuring new features is documented.

Configure Token Lifetime Properties

Customize token lifetime properties in the #OTK Token Lifetime Configuration policy.



NOTE

Temporary tokens have short lifetimes for security reasons. We recommend you do not extend the default value for temporary tokens.

To configure token lifetimes:

1. In the Policy Manager, navigate to OTK/PolicyFragments/configuration.
2. Open the OTK Token Lifetime Configuration policy to view the token lifetime default settings. Click **Show Comments** to see helpful translations from seconds to minutes or hours.
3. Copy the assertions of any Set Context Variables you want to customize.
4. Open the #OTK Token Lifetime Configuration policy found in the OTK/Customizations folder.
5. Paste the assertions and set new values for the Context Variables.

6. Save and Activate.

The following guidelines apply:

- Set the `access_token` to expire before the `refresh_token`.
- The `consumer_key` and `client_id` values can be set to 0 (never expire).
- To effectively have a token that does not expire, set the value to the maximum value of 631138520 seconds (20 years).

OAuth 1.0 Variables	Description
<code>oauth_v1_consumer_key_lifetime_m</code>	Controls the lifetime of OAuth consumer keys. Default: 0 minutes (never expire)
<code>oauth_v1_access_token_lifetime_s</code>	Controls the lifetime of access tokens. Set to 0 to make the token invalid immediately. Default: 86400 seconds = 1 day.
<code>oauth_v1_request_token_lifetime_s</code>	Controls the lifetime of request tokens. Set to 0 to make the token invalid immediately. Default: 300 seconds = 5 minutes.
OAuth 2.0 Variables	
<code>oauth2_auth_code_lifetime_sec</code>	Controls the lifetime of issued OAuth codes. Set to 0 to make the code invalid immediately. Default: 300 seconds = 5 minutes
<code>oauth2_access_token_lifetime_sec</code>	Controls the lifetime of issued access tokens. Set to 0 to make the token invalid immediately. Default: 3600 seconds = 1 hour
<code>oauth2_refresh_token_lifetime_sec</code>	Controls the lifetime of issued refresh tokens. Default: 604800 seconds = 1 week To effectively have a token that does not expire, set the value to the maximum value of 631138520 seconds (20 years).
<code>oauth2_client_id_lifetime_m</code>	Default: 0 minutes (never expire)
<code>oauth2_client_id_lifetime_SDK_m</code>	Default: 10080 minutes = 1 week
OpenID Connect Variables	
<code>id_token_lifetime_s</code>	The lifetime of the OpenID session. Default: 86400 seconds = 1 day

Client-Specific Customization

In OAuth Manager, you can configure settings for OAuth Clients and Client Keys through the Custom Field. Expressed as a JSON message, this configuration is stored in the `#{custom}` variable. You can then use this variable for client specific configuration within policy.

This page contains the following sections:

Set Parameter Values in the Custom Field for a Client (or Client Key)

To set parameter values in the Custom JSON Field for a client or client key:

1. Open a browser and go to: `https://<yourgatewayURL>:8443/instanceModifier/oauth/manager`
2. Provide a username and password. The type of access you are granted depends on your user role.
3. Click **Clients**.
4. Select a client and click **Edit**.
The Edit page appears with a **Custom Field** that accepts JSON content.
5. Provide JSON content that configures parameters for this client.

6. Click **Update Client**.

Similarly, you can click **List Keys** for a client, then **Edit**. Provide custom JSON values for the client key and click **Save**.

For example:

Custom Field
(MUST be JSON or empty)

```
{
  "lifetimes": {
    "oauth2_access_token_lifetime_sec": 86400,
    "oauth2_refresh_token_lifetime_sec": 432000
  }
}
```

How the Custom Field Content is Stored

The custom field contents are stored in the `#{custom}` variable that has the following structure:

```
{"client_custom": #{client_custom}, "client_key_custom": #{client_key_custom}}
```

Add Custom Logic to Extend the #Policy

Refer to the examples for the nesting logic.

Tasks include:

- Creating a `custom_json` Context variable to hold the JSON message content provided by the `#{custom}` variable.
- Using an Evaluate JSON Path Expression assertion to extract the key/value pairs.
- Adding a Compare Expression assertion to check if any custom values were set.
- Overwriting the default setting for the client with the custom values.

Set Context Variable: `custom_json`

Add a Set Context Variable assertion called `custom_json` to hold the content of `#{custom}`. The `#{custom}` variable contains the JSON message parameter settings for the client and client key.

Use the following settings for the Set Context Variable assertion:

Variable Name – `custom_json`

Data Type – Message

Content-Type – `application/json; charset=UTF-8`

Expression – `#{custom}`

Evaluate JSON Path Expression

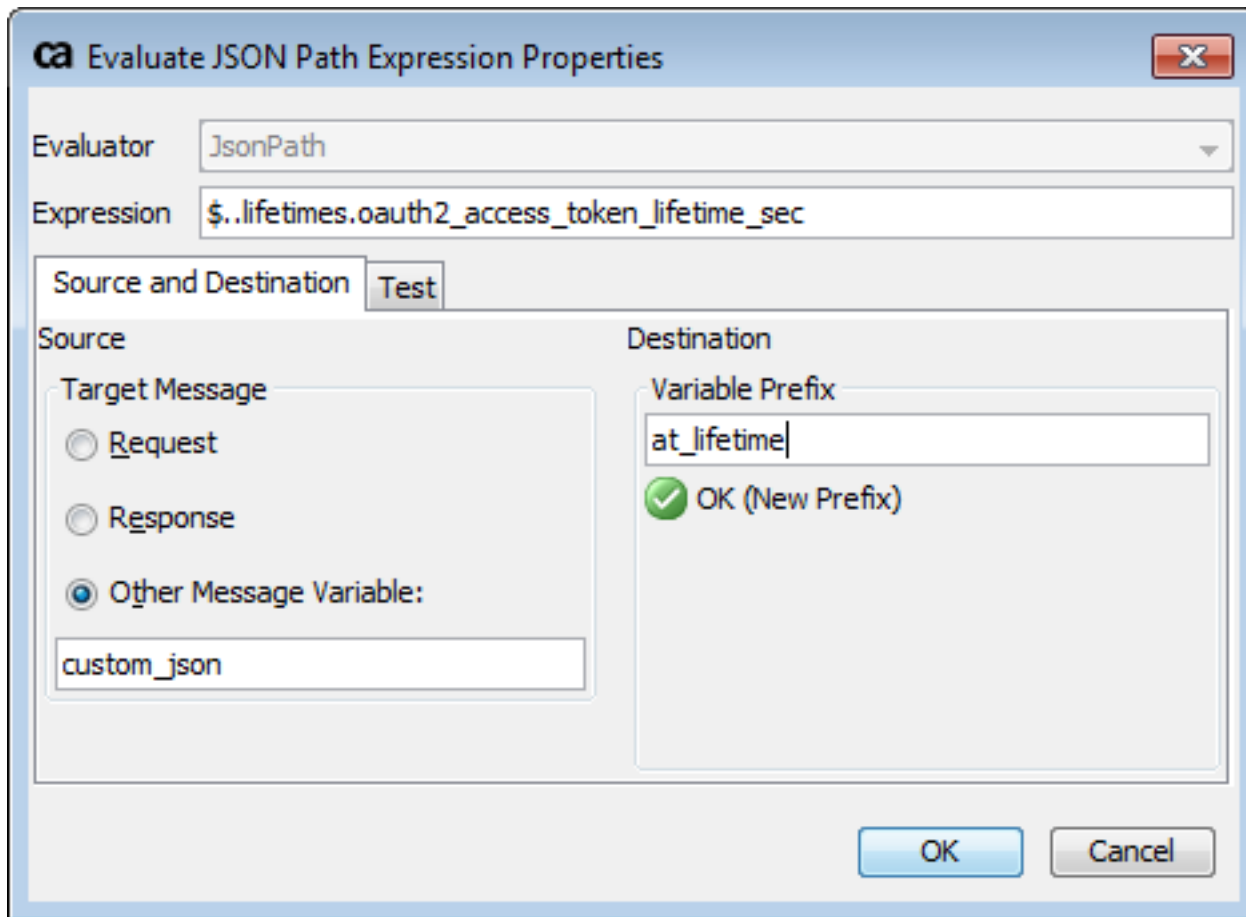
Set up the Evaluate JSON Path Expression to capture the parameters.

Use the following settings for the Evaluate JSON Path Expression assertion:

Expression – Identify the parameters you want to extract. For example: `$.lifetimes.oauth2_access_token_lifetime_sec`.

Source – Click Other Message Variable and type `custom_json`. This identifies where to find the custom content.

Destination – Create a new variable prefix to identify the result of the extraction. For example: `at_lifetime` becomes `at_lifetime.result` and holds the custom access token lifetime value.



Add a Compare Expression Assertion

Add a Compare Expression assertion to check if any custom values are set. The expression is specific to the parameters you are checking. For example:

Use the following settings to check to see if the "lifetimes" element exists in the JSON message:

Variable – `custom_json`

Data Type – Unknown/Other

If Multivalued – All values must pass

Add – Simple Comparison

Set up the Simple Comparison rule as:

(does) **contain Right Expression** – `lifetimes` (or whatever you're checking here for content).

Case Sensitive – unclick

Overwrite the Default Setting of the Context Variable

Add a Set Context Variable assertion to overwrite the parameter with the result of the JSON extraction. The result of the JSON extraction is stored in a variable created with the assigned Variable Prefix in the following format: `variablePrefix.result`.

Hover your mouse over the **Evaluate JSON Path Expression** assertion to see what variables are automatically created.

In the following example, the Variable Prefix is added to the `.found`, `.count`, and `.result`, and results.

`$(custom_json): Evaluate JSON Path Expression - ..lifetimes.oauth2_access_token_lifetime_sec`

Sets `$(at_lifetime.found)`, `$(at_lifetime.count)`, `$(at_lifetime.result)`, `$(at_lifetime.results)`

The Set Context Variable has the following properties:

Variable Name – The Context Variable you are customizing for this client. For example:

`oauth2_access_token_lifetime_sec`

Expression – The variable where the extracted value from the JSON message is stored. For example:

`$(at_lifetime.result)`

Client-Specific Customization Examples

The following examples show how client-specific customization can be implemented:

Customizing Token Lifetime for a Specific Client

The following policy example shows how custom settings for the access token lifetime and refresh token lifetime are set for a specific client. The configuration is performed in the **#OTK Token Lifetime Configuration** policy.

- ▼ custom At least one assertion must evaluate to true Search for custom token lifetimes
 - ▼ extract All assertions must evaluate to true Extract the token lifetime values from the custom JSON message
 - ✓ JSON Set Context Variable custom_json as Message to: `$(custom)` Turn the custom string into a JSON message
 - ✓ lifetimes Compare Variable: `$(custom_json.mainpart)` contains lifetimes; If Multivalued all values must pass No lifetimes specified, continue
 - `access_token` `$(custom_json): Evaluate JSON Path Expression - $..lifetimes.oauth2_access_token_lifetime_sec`
 - `refresh_token` `$(custom_json): Evaluate JSON Path Expression - $..lifetimes.oauth2_refresh_token_lifetime_sec`
 - ✓ override Set Context Variable `oauth2_access_token_lifetime_sec` as String to: `$(at_lifetime.result)` Override the pre-configured `access_token` lifetime variable
 - ✓ override Set Context Variable `oauth2_refresh_token_lifetime_sec` as String to: `$(rt_lifetime.result)` Override the pre-configured `refresh_token` lifetime variable
 - ✓ Continue Processing

Customizing Token Behavior for a Specific Client Identified by Client Key

The following example shows how the **#OTK Storage Configuration** policy was extended to provide global defaults and custom token behavior for a specific client key.

The logic includes:

- Global configuration by setting default values for the following Context Variables:
 - `max_oauth_token_count`** = 5
 - `max_oauth_token_behaviour`** = error
- Specific configuration for clients/client keys that overrides the global configuration:
 - For the Client Key, the following custom values are set:

Custom Field

(MUST be JSON or empty)

```
{
  "max_token_count" : 10,
  "max_token_behavior" : "cycle"
}
```

- Similarly, the following custom values are set for the client:
{"max_token_count": 7, "max_token_behaviour": "cycle"}
- This configuration is not applicable to **CLIENT CREDENTIALS** grant type.

The policy can be coded as follows:

```

2  Comment: Target Configuration Policy: "OTK Storage Configuration"
3  Comment: === Set custom values for Context Variables below ===
4  Global Token Count All assertions must evaluate to true
5  ✓ Set Context Variable max_oauth_token_count as String to: 5
6  ✓ Set Context Variable max_oauth_token_behaviour as String to: error
7  Comment: === Add any new Context Variables or extensions below ===
8  Per Client Token Count At least one assertion must evaluate to true
9  All assertions must evaluate to true
10 ✓ Set Context Variable custom_json as Message to: ${custom}
11 At least one assertion must evaluate to true
12 Client Key Custom All assertions must evaluate to true
13   [E] ${custom_json}: Evaluate JSON Path Expression - $..clientkey.client_key_custom.max_token_count
14   ✓ Compare Variable: ${token_count.count} is greater than 0 (case sensitive); If Multivalued all values must pass
15   ✓ Set Context Variable max_oauth_token_count as String to: ${token_count.result}
16 Client Custom All assertions must evaluate to true
17   [E] ${custom_json}: Evaluate JSON Path Expression - $..clientkey.client_custom.max_token_count
18   ✓ Compare Variable: ${token_count.count} is greater than 0 (case sensitive); If Multivalued all values must pass
19   ✓ Set Context Variable max_oauth_token_count as String to: ${token_count.result}
20 ✓ Continue Processing

```

Global Token Count

The global section sets the rule. It contains Context Variables that establish default values for all clients. There is no check required for the `${custom}` variable.

The code determines the following global behavior:

- With the **max_oauth_token_count** set to 5, all clients can access four additional instances of the same app without logging out of the first instance.
- When a client attempts to log into more than five instances, the **max_oauth_token_behaviour** setting indicates that an error is returned.

Per-Client Token Count

The per-client section sets the exception to the rule. It contains logic that checks for any content within the `#{custom}` variable, extracts the values of parameters associated with the client and client key, then overrides the default global setting of these parameters for the specific client.

The code to set per-client behavior is as follows:

- The `custom_json` Context Variable holds the `#{custom}` variable contents.
- The JSON contents are extracted.
- If the `max_oauth_token_count` is found in the JSON object, the Context Variable `max_oauth_token_count` is set to its associated value.

In this example, because the **Client Custom** and **Client Key Custom** fields are both set, the **Client Key Custom** takes precedence as it is the first assertion code block found to be true.

Configure the Authorization Server

The web content for the authorization server is hosted in the OTK Authorization Server Website Template policy. Customize the website by changing the text, the logo, and style sheet. Do not change the variable names or remove variables.

Alternatively, host the website on an external web server and point to the location.

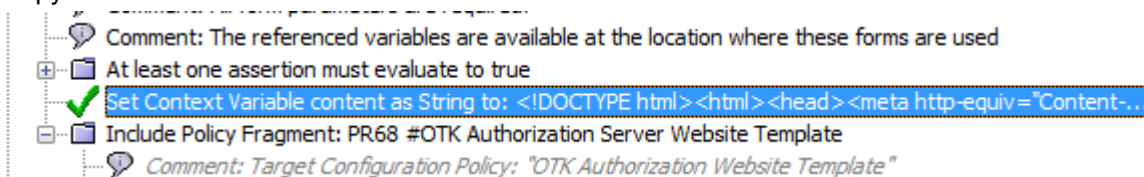
Customize Graphic Elements

Graphic elements appear in the header and footer of the website page and include the corporate logo and style sheet. The default logo is "CA Technologies".

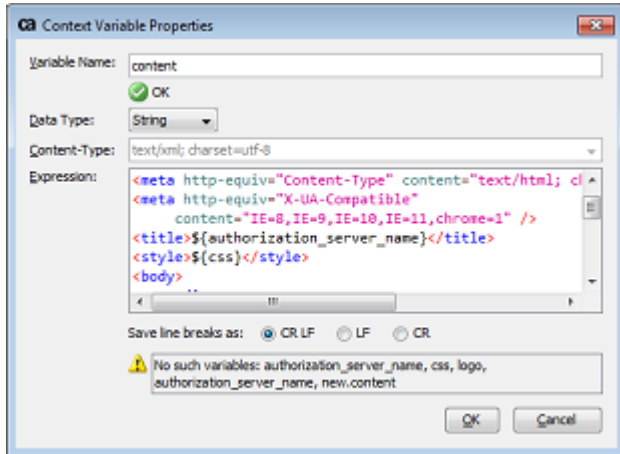


To customize graphic elements of the authorization server website template:

1. In Policy Manager, navigate to OTK/Policy Fragments/configuration/Authorization Server.
2. Open the **OTK Authorization Server Website Template** policy and expand the assertions. You cannot edit this read-only policy directly.
3. Copy the **Set Context Variable content** assertion. The assertion contains an HTML form.



4. Navigate to OTK/Customizations/Authorization Server/
5. Open the **#OTK Authorization Server Website Template** policy and locate the following comment: "getting started: copy variable 'content'..."
6. Paste the Set Context Variable content assertion.
7. Double-click the Set Context Variable content assertion to access the HTML content.



8. Expand the dialog box and edit the following HTML content by providing custom information for the following elements:

Element	Default	Notes
<title>	\${authorization_server_name}	The name of the web page
<style>	\${css}	Provide your internal style sheet HTML, or replace the <style> element with <link> and reference one or more external style sheets. For example, <pre><link rel="stylesheet" type="text/css" href="mystyle.css"></link></pre>
	\${logo}	Provide the data URI or a link for your custom logo. For example: <pre></pre>
<div id="dynamicContent">	\${new.content}	Replace with your custom website template content.
<p class="portal-copyright">	@ CA Technologies. All rights reserved.	Replace the text with your copyright information.

9. Click **OK** to save the customization changes.

10. **Save and Activate** the #policy.

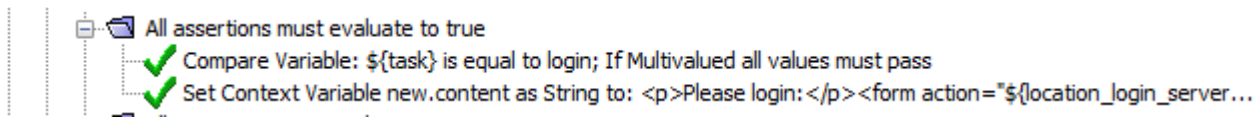
Customize Text Content

Default text shown on the Authorization Server webpage is associated with the following three tasks:

- Login
- Consent
- Reset

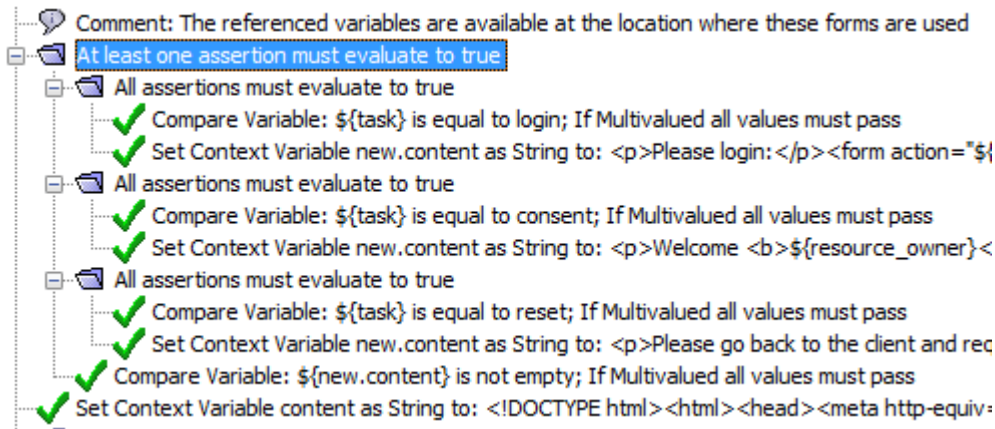
The text for these tasks is expressed in a new.content variable and is associated with a corresponding \${task} variable. For example, the variable \${task} equal to login

is followed by the new.content Context Variable containing the "Please login" string.



To customize the text:

1. In Policy Manager, navigate to OTK/Policy Fragments/configuration/Authorization Server.
2. Open the **OTK Authorization Server Website Template** policy.
3. Within the policy, find the folder that contains the three assertions that set the **new.content** variable.
4. Select the folder and copy it.



5. Open the **#OTK Authorization Server Website Template** policy located in OTK/Customizations/Authorization Server.
6. Paste the folder containing the assertions directly following:
 Comment === Add any new Context Variables or extensions below ===
 The code must appear before the logic that sets the website template formatting.
7. Double-click the assertions and modify content for any of the `${new.content}`.
8. **Save and Activate**.

Default Content

Task	Default Content
login	<pre> <p>Please login:</p> <form action="{location_login_server}" method="POST" class="form-body form- login" style="margin: 0;"> <input type="hidden" name="sessionID" value="{sessionID}"/> <input type="hidden" name="sessionData" value="{sessionDataJWT}"/> <div class="control-group"> <label>Username *</label> <input type="text" name="username" class="input-block"> </div> <div class="control-group"> <label>Password *</label> <input type="password" name="password" class="input-block"> </div> <div class="row-fluid"> <div class="span12"> <button type="submit" class="btn btn- primary pull-right" name="action" value="login" style="margin-left: 2em;">Login</button> <button type="submit" class="btn btn- primary pull-right" name="action" value="cancel" style="margin-left: 2em;">Cancel</button> </div> </div> </form> <div style="clear: both; padding-top: 1em;"> <!--social_login--> </div> </pre>

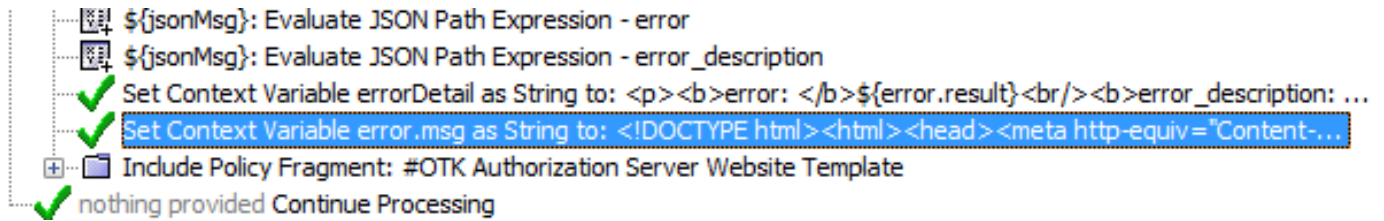
consent	<pre> <p>Welcome \${resource_owner} (not \${resource_owner}? Click here</p> <p> A client with the following properties is seeking access to resources: </p> <table> <tr><td>Client Name:</td><td> \${client_name}</td></tr> <tr><td>SCOPE (permissions):</td></tr> <tr><td>\${scope}</td></tr> </table> <p>Please grant or deny the request</p> <form action="\${location_consent_server}" method="POST"> <input type="hidden" name="sessionID" value="\${sessionID}"/> <input type="hidden" name="sessionData" value="\${sessionDataJWT}"/> <table> <tr> <td><button type="submit" class="btn btn-primary pull-right" name="action" value="Deny" style="margin-left: 2em;">Deny</button></td> <td><button type="submit" class="btn btn-primary pull-right" name="action" value="Grant" style="margin-left: 2em;">Grant</button></td> </tr> </table> </form> </pre>
wrong user	<pre> <p>Please go back to the client and request access to resources again</p> </pre>

Customize the Error Message

The error message is set as HTML code in the error.msg Context Variable in the OTK Authorization Server Website Template policy.

To customize the error message:

1. In Policy Manager, navigate to OTK/Policy Fragments/configuration/Authorization Server.
2. Open the **OTK Authorization Server Website Template** policy.
3. Locate and copy the Set Context Variable error.msg assertion that has default HTML content assigned.



4. Open the **#OTK Authorization Website Template** policy located in OTK/Customizations/Authorization Server.
5. Find the comment: `===getting started: copy 'error.msg'...`
6. Paste the Set Context Variable `error.msg` assertion.
7. Double-click the assertion and provide your custom error message in HTML format.
8. Click **OK**.
9. **Save and Activate**.

Host Pages on External Servers

By default, the Authorization server is hosted on the Gateway server (localhost). Optionally, you can use an external website to host the login and consent pages. Different servers can be used for login and consent. Integrate with an existing login page, or create a new login page.

Any external authorization server must be able to perform the following tasks:

- Validate and create signed & encrypted JWT using a shared secret.
- Authenticate the user.
- Redirect to `/auth/oauth/v2/authorize/consent` and include all required parameters.

To host the website on an external web server:

1. In Policy Manager, navigate to OTK/Policy Fragments/configuration/Authorization Server.
2. Open the **OTK Authorization Server Configuration** policy to access context variables shown in the following table.

Context Variable	Default Value	Notes
host_login_server	<code>https://\${gateway.cluster.hostname}:8443</code>	The hostname of the login server
host_consent_server	<code>https://\${gateway.cluster.hostname}:8443</code>	The hostname of the consent server
path_login_server	<code>/auth/oauth/v2/authorize/login</code>	The path to the endpoint receiving the login request
path_consent_server	<code>/auth/oauth/v2/authorize/consent</code>	The path to the endpoint receiving the consent request

3. Copy the Set Context Variable assertions for any Context Variable you want to modify.
4. Open the **#OTK Authorization Server Configuration** policy.
5. Paste the Set Context Variable assertions and provide custom values for hostname and path information.

NOTE

For more information about integrating with an external server, see the following blog post:








<https://communities.ca.com/blogs/oauth/2016/10/04/howto-integrating-otk-with-external-login-server>

Secure Client/Server Communication

NOTE

Prior to OTK versions 4.2, custom configuration settings for Authorization server communication were made in the **OTK Authorization Server Configuration** encapsulation assertion. This customization no longer takes effect. Copy any previous customization made in **OTK Authorization Server Configuration** into the **OTK Security Header Extension** policy.

The **OTK Security Header Extension** policy contains optional security settings that ensure HTTPS communication and prevent clickJacking attacks. By default, these security settings are turned off. The assertions are disabled.

-  Comment: These HTTP headers will always be added to a response from all services
-  Comment: Authorization Server and OAuth Manager Security Header Extension policies will override values set here for their respective services
-  Response: Add HTTP Header Content-Security-Policy:default-src 'self'; script-src *.googleapis.com 'unsafe-inline'; img-src * data:; style-src 'unsafe-in
-  Response: Add HTTP Header X-Content-Type-Options:nosniff (replace existing)
-  Response: Add HTTP Header X-XSS-Protection:1 (replace existing)
-  Response: Add HTTP Header Strict-Transport-Security:max-age=31536000; includeSubDomains (replace existing)
-  Response: Add HTTP Header X-Frame-Options:Deny (replace existing)

To enable and customize client/server communication settings:

1. Navigate to OTK/Customizations
2. Open the **OTK Security Header Extension** policy.
3. Enable any of the disabled assertions. The default settings when these assertions are enabled are described below.
4. Optionally customize the Context Variables and click **OK**.
5. **Save and Activate**.

Security	HTTP Header Default Value	Notes
Enforcement of HTTPS	Strict-Transport-Security : max-age=31536000; includeSubDomains (replace existing)	Responses include the "Strict-Transport-Security" header (HSTS) that restricts browser communication to HTTPS only. For more details on HSTS, see RFC6797 . The max-age parameter is the time in seconds after the initial reception that the HTTPS policy is enforced for the host. If the host attempts an HTTP communication within this time, it is rejected. Default value is one year. The includeSubDomains parameter adds subdomains of the host to the HTTPS restriction policy.
Enforcement of no frame overlap	X-Frame-Options :Deny (replace existing)	Protects browsers from clickJacking attacks by preventing overlapping of multiple frames. Options include: Deny – The page cannot be displayed in a frame. Sameorigin – The page can only be displayed in a frame on the same origin as the page itself.

Configure PKCE Support

Proof Key for Code Exchange (PKCE) is supported for enhanced authorization code security. By including a code challenge to the authorization flow, it addresses the case where an authorization code is intercepted as it is sent back to the client. For more information on the PKCE protocol and the security considerations, see [IETF RFC 7636](#).

NOTE

Use of PKCE is optional. If your client request does not include a PKCE code challenge, the normal authorization flow is followed.

However, device registration with an authorization code requires the use of PKCE.

For an authorization request to use PKCE:

1. The client creates a one-time secret (**code verifier**) that is used to generate a **code challenge**.
2. The client sends the **code challenge** value with the Authorization Request to the Authorization Server.
3. The server saves the **code challenge** and returns the Authorization Code to the client.
4. The client sends the Authorization Code and the **code verifier** to the server to get an access token.
5. The server checks the Authorization Code, but also the validity of the request source by using the **code verifier** to create a **code challenge**.

If the **code challenge** value matches the previously stored **code challenge**, the request source is validated. The server returns an access token.

If no **code verifier** is provided, or the **code challenge** values do not match, no access token is returned.

A malicious app using an intercepted Authorization Code cannot generate a matching code challenge and is, therefore, not granted an access token.

OTK PKCE Validation Encapsulated Assertion

The following endpoints for authorization include the OTK PKCE Validation encapsulated assertion:

Endpoint	Notes
/auth/oauth/v2/authorize	Checks that code_challenge is present. If present, the code_challenge_method must exist. Adds code_challenge and code_challenge_method to the session created with each authorization request.
/auth/oauth/v2/token	Uses the authorization_code to look up the associated session. If code_challenge exists, code_verifier must be provided as an input. Recalculates the code_challenge, from the code_verifier value using the persisted code_challenge_method. Compares the calculated code_challenge to the persisted code_challenge.
/connect/device/register	Checks that code-verifier exists in the header when an authorization code is used to register a device. Used with social login and device2device login.

The following PKCE parameters are passed in by client in the URL of an authorization request.

PKCE Parameters	Notes
code_verifier	A random value of 43-128 characters. Created by the client. Stored on the server upon an authorization request. The recommended value is a 32-octet sequence that is base64url-encoded to create a 43-octet URL safe string.

code_challenge_method	The code challenge method used to generate the code challenge value. One of the following: <ul style="list-style-type: none"> • plain – The code verifier value is used as the code challenge value. This method is used when SHA256 encoding is not available. • S256 – SHA256 encoding is performed: code_challenge=base64url(sha256(code_verifier))
code_challenge	Value based on the code_verifier and the code_challenge_method.

The following 112 error code is related to PKCE:

```
{
  "error": "invalid_request",
  "error_description": "The given code_challenge or code_challenge_method is invalid"
}
```

Provide Enhanced HTML Form Security

Use any of the following procedures to address known security issues with the submission of HTML forms.

Use the "Protect Against Code Injection" Assertion

By default, the "Protect Against Code Injection" assertion inhibits the submission of HTML tags in the HTML-form data. However, you can restrict any of the following additional character classes:

- PHP
- Shell
- LDAP
- DN
- LDAP Search
- XPath injection

Be aware that restricting characters in form data can cause problems. For example, restricting DN would cause the submission of a callback URL of the format: `https://<domain>:<port>/`.

Disable Auto-Complete for the OTK Authorization Server

By default, HTML form auto-complete is enabled. This common browser behavior allows for quicker entry of usernames and passwords.

However, you may wish to disable auto-complete on the OTK Authorization Server Website Template for improved security.

NOTE

HTML form auto-complete is used in the OTK Authorization Server Website Template, the OAuth Manager, and the Test Clients. However, the OAuth Manager and Test Clients are for internal use only. Consequently, HTML forms for these they do not typically require the extra security provided by disabling auto-complete.

Turn auto-complete off by editing the `<form>` element and adding the `autocomplete=false` parameter setting.

OTK Authorization Server Website Template

To disable form auto-complete:

1. Access **OTK Authorization Server Website Template** at OTK/Policy Fragments/configuration/Authorization Server/
2. Search for "please login". This phrase appears in the first Set Context Variable assertion for the **new.content** variable.
3. Copy the assertion.
4. Open **#OTK Authorization Server Website Template** policy found in OTK/Customizations/Authorization Server/
5. Paste the assertion.
6. Double click the assertion and, within the `<form>` element, add `autocomplete=off`

```
<p>Please login:</p>
```

```
<form action="{location_login_server}" method="POST" class="form-body form-login" style="margin: 0;" autocomplete="off">
```

7. Click **OK** to close the assertion properties dialog.
8. **Save and Activate.**

Customize Caches

OTK policies take advantage of caching to avoid database calls and improve performance. The policies use local caches (visible on a single node only) and database-backed caches (visible throughout all cluster nodes). The default configuration for these caches is optimized for performance. We recommend using the default settings.

Customization of cache properties is more likely when you create caches for your own services.

The following sections relate to OTK cache customization:

OTK Cache Encapsulated Assertions

The following OTK encapsulated assertions are used in multiple policies:

- OTK Cache Store
- OTK Cache Look Up
- OTK Cache Remove
- OTK Cache Flush

OTK Cache Encapsulated Assertion	Required Input Variables	Optional Input Variables	Notes
OTK Cache Store	cacheID cacheKey	maxEntriesmaxEntryAge maxEntrySize	Stores an item to the cache. If the cache does not exist, one is created. If the key exists, the existing item is overwritten. Reduces the load on back-end services and improves response times. Optional Variables: maxEntries – The number of cached entries that the store can hold. When this maximum is reached, each new item replaces the oldest one in the store. maxEntryAge – The maximum age (in seconds) of items in the cache before they are discarded. maxEntrySize – The maximum size (in bytes) of the items to cache.
OTK Cache Look Up	cacheID cacheKey	maxEntryAge	Retrieves the cached contents from an existing cache store. On success, the contents are placed into the message target of your choice (request, response, or context variable). On failure, an error is returned. The maxEntryAge variable acts as a filter. If an item is determined to be below this age, it is retrieved from the cache and returned in the response. If the cached item exceeds this age, it is not retrieved.
OTK Cache Remove	cacheID cacheKey	maxEntriesmaxEntryAge maxEntrySize	Removes an item from the cache. Items meeting criteria set by any of the optional properties are removed. Items exceeding any of the maximum thresholds set by optional properties remain in the cache.
OTK Cache Flush	cacheID	maxEntryAge	Empties the cache. If an item is determined to exceed the maxEntryAge value, it remains in the cache.

To learn how the OTK Cache Look Up and OTK Cache Store assertions are used together, examine existing policies such as OTK Client Validation.

OTK Cache Remove and OTK Cache Flush assertions are used for the specific removal of items and cache clean up.

What are the Default Cache Settings?

Default settings for the OTK cache assertions are set in the read-only OTK Cache Handler policy.

The following input variables are set with default values and are used as criteria for storage, retrieval, and removal operations:

- `maxEntries` – The default is 10000.
Indicates the number of cached entries that a store can hold. When this maximum is reached, each new item replaces the oldest one in the store.
- `maxEntryAge` – The default is 300 seconds.
Indicates the maximum age (in seconds) of items in the cache before they are discarded.
- `maxEntrySize` – The default is 10000 bytes.
Indicates the maximum size (in bytes) of the items to cache.

The Cache Handler policy and the OTK encapsulated assertions are located in OTK/Policy Fragments/caching.

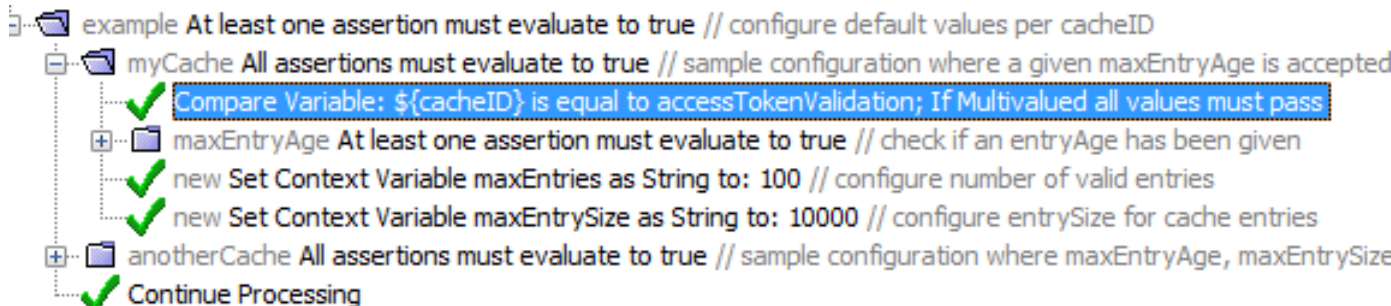
How do I Customize Settings for a Specific Cache?

The OTK Cache Handler policy is read-only. To set custom cache settings, set context variables in the **OTK Caching Customization** policy.

The combination of `cacheID` and `maxEntryAge` values create the `actualCacheID` variable that is used by the look up and store cache operations.

To customize a cache:

1. In Policy Manager, go to OTK/Customizations/caching.
2. Open the OTK Caching Customization policy.
The policy contains disabled folders providing a template for each cache you want to customize.
3. Expand the folders, right-click, and select **Enable All Assertions**.
4. Click **Show Comments** to view the example.
5. In the block of assertions for `myCache`, double-click the "Compare Variable: `#{cacheID}` is equal to `myCache`: If Multivalued all values must pass" assertion.
6. Click **Edit** and replace the Right Expression "`myCache`" default value with the `cacheID` of the cache you want to customize.
For example, "`accessTokenValidation`".

7. 

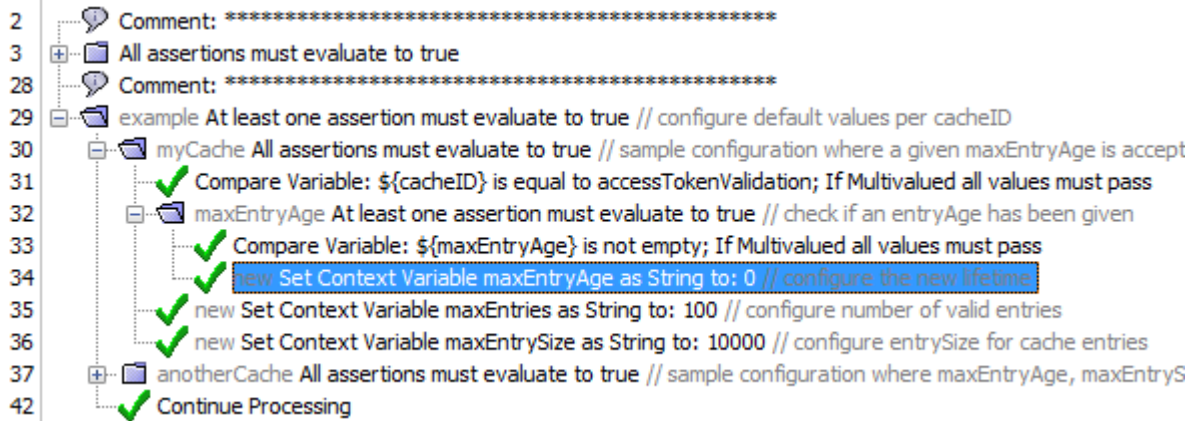
```

example At least one assertion must evaluate to true // configure default values per cacheID
├── myCache All assertions must evaluate to true // sample configuration where a given maxEntryAge is accepted
│   ├── Compare Variable: #{cacheID} is equal to accessTokenValidation; If Multivalued all values must pass
│   ├── maxEntryAge At least one assertion must evaluate to true // check if an entryAge has been given
│   ├── new Set Context Variable maxEntries as String to: 100 // configure number of valid entries
│   └── new Set Context Variable maxEntrySize as String to: 10000 // configure entrySize for cache entries
├── anotherCache All assertions must evaluate to true // sample configuration where maxEntryAge, maxEntrySize
└── Continue Processing

```

Click **OK**.

8. Provide custom values for one or more Context Variables.
The screenshot shows customization of the `accessTokenValidation` cache with the `maxEntryAge` Context Variable set to zero (0). This setting requires a database lookup for each single `access_token` validation.



9. Remember to delete any Context Variables that you do not want to customize!
The Context Variable settings in the OTK Caching Customization policy override the default settings in the policies using the cache.
10. **Save and Activate.**

Monitoring Caches

Use the Add Audit Detail assertion to monitor content going into and retrieved from the cache.

The messages are recorded either in the audit records or a Gateway log, depending on how the assertion is configured. If audit details are directed to the audit log, the message appears in the "Associated Logs" tab in the Event Details Pane of the Gateway Audit Events window.

To monitor content going into the cache:

1. Go to OTK/Customizations/caching and open the **OTK Cache Store Customization** policy.
2. Place an **Add Audit Details** assertion before the **Store to Cache** Assertion.
3. Double-click the Add Audit Details assertion and add the properties you want to return. See [Configuring Audit Detail Properties](#).
4. **Save and Activate.**

To monitor content retrieved from the cache:

1. Open the OTK Cache Lookup Customization policy
2. Place an **Add Audit Details** assertion after of the **Look Up in Cache** assertion.
3. Double-click the Add Audit Details assertion and add the properties you want to return. See [Configuring Audit Detail Properties](#).
4. **Save and Activate.**

Configuring Audit Detail Properties

ca Audit Detail Properties

Message: `CUSTOMIZE : actualID = ${actualCacheID} ; cacheKey = ${cacheKey} ; max entries = ${maxEntries}; max entry age = ${maxEntryAge}; max entry size = ${maxEntrySize}`

Category: Audit Log Custom logger name: `com.l7tech.log.custom.`

Level: **WARNING** ▼

OK Cancel

Setting	Description
Message	Type a message in the box. This message is displayed when the audit appears in the Gateway Audit Events window. Include context variables within the message to reveal additional information about the audit condition, if necessary.
Audit	Select this option to direct the audit detail message to the Audit log sink.
Log	Select this option to direct the audit detail message to the Gateway log sink. This is useful for situations where (for example) the logged information is too large to be comfortably stored in the audit database for extended periods of time. For example, storing trace information from a policy debug tracing.
Custom logger name	Select this check box if you want the logged information to be identified by a custom logger name, rather than the default logger name <code>com.l7tech.server.policy.assertion.ServerAuditDetailAssertion</code> . If you choose to use a custom logger name, enter a suffix to be added to the custom logger name, to ensure uniqueness. You can reference context variables. If a specified context variable cannot be resolved during run time, the default logger name shown above is used.
Level	Select a severity level for your message from the drop-down list. This level, along with the level set in the Audit Messages in Policy assertion, determines whether your message appears in the Gateway Audit Events window.

Existing Caches and Policies

For your reference, the following caches are used by read-only OTK policies and target endpoints.

Cache ID	Target Policy or Endpoint Using this Cache	Notes
allClientValuesCache	OTK Client DB Get OTK Client DB Revoke Key Client NoSQL GetOTK Client NoSQL Revoke Key OTK Client NoSQL Update	OTK Client DB Update OTKs when access_token requests are made. The maxEntryAge value determines how often the client configuration is looked up from the database. Reduce the cache maxEntryAge value for more frequent calls. Increase the value for less frequent calls.
allClientValidationValuesCache	OTK Client DB Revoke Key OTK Client DB Update OTK Client NoSQL Revoke Key OTK Client NoSQL Update OTK Client Validation	OTK Client Validation is used to validate that the client is still active in the system. By default, the allClientValidationValuesCache stores the following client details: client_type client_name scope environment created_by client_custom client_key_custom serviceids accountplanmappingids client_secret
defaultCache	OTK Session DB	Caches the OTK session information. The cache name can be configured when using any encapsulated assertion named "OTK Session".
openIDConnectCache	OTK Session GET	Used to retrieve values associated with an authorization_code in the context of OpenID Connect when the client exchanges the code for an access_token. OTK grant_type=AUTHORIZATION_CODE

Cache ID	Target Policy or Endpoint Using this Cache	Notes
accessTokenValidation	OTK Require OAuth 2.0 Token	<p>Caches access_token validation results at OAuth 2.0 protected endpoints. This value is a compromise between performance and accepted lifetime of invalid tokens. The default cache time for saving client information is set to 30 seconds. Increasing this time elevates the risk of unauthorized access.</p> <p>The lifetime value is passed in through the interface of the encapsulated assertion wherever access_token_validation is used. Modify the cache lifetime to lookup access_tokens from the database more or less frequently.</p> <p>A lifetime value of 0 reduces performance by requiring a database lookup for each single access_token validation. Increasing the lifetime value extends the expiration time of an access_token.</p>
authorizeCache	/auth/oauth/v2/authorize	<p>Caches the static content on the authorization server website (except for images). This is useful if the website is hosted on an external web server.</p> <p>If the website template for the authorization server website changes often, consider modifying the cache lifetime. The cache does not include form values of images.</p>
userSessionIDCacheV2	/auth/oauth/v2/authorize OTK Session - Store	Used with the session cookie "I7otk2a" Customize the lifetime of the user session (cookie lifetime) by modifying the variable "ownerCacheAge. The value is expressed in seconds.
I7ManagerCache	/oauth/manager /oauth/manager/clients /oauth/manager/tokens OTK Session - Store	Used with the cookie "I7manager".
consumerSecretCache	/oauth/validation/validate/v1/signature	Used with temporary values during the OAuth 1.0 flow to validate the signature.

Set an Alternative HTTPS Port

By default, the OAuth Toolkit policies are configured to use Port 8443 for HTTPS communication. This includes logging in to the OAuth Manager.

Customize the OTK to use an alternative port by configuring both policies and APIs:

Configuring Policies

The following policies contain references to Port 8443:

Custom configuration is made in the corresponding #policies.

NOTE

In each case for the policies you:







1. Copy the default Set Context Variable assertions containing the 8443 port from the read-only policy.
2. Paste the default assertions into the corresponding editable #policy.
3. Edit the port number.
4. Save the #policy.

OTK Client Context Variables

This policy contains three references to the default 8443 port.

To set an alternate port number:

1. In Policy Manager, open the read-only **OTK Client Context Variables** policy. This policy contains context variables set to the default 8443 port.
2. Select and copy the following Set Context Variable assertions in the read-only policy. Use Ctrl-click to select multiple assertions.
 - host_oauth2_auth_server
 - host_oauth_manager
 - host_oauth_test_clients
3. Open the **#OTK Client Context Variables** policy found in the Customizations/Tools folder. By default, no context variables are set.
4. Paste the copied assertions into the **#OTK Client Context Variables** policy.
5. Double-click each assertion, edit the port number, and click **OK**.
6. **Save and Activate** the #OTK Client Context Variables policy.

-  Comment: Target Configuration Policy: "OTK Token Lifetime Configuration"
-  Comment: === Set custom values for Context Variables below ===
-  Comment: === Add any new Context Variables or extensions below ===
-  Set Context Variable host_oauth2_auth_server as String to: https://{gateway.cluster.hostname}:9443
-  Set Context Variable host_oauth_manager as String to: https://{gateway.cluster.hostname}:9443
-  Set Context Variable host_oauth_test_clients as String to: https://{gateway.cluster.hostname}:9443

OAuth Manager Config

This policy contains one reference to the default 8443 port.

To set an alternate port number:

1. In Policy Manager, open the read-only **oauth manager config** policy.
2. Copy the Set Context Variable assertion for this.app.url:
Set Context Variable this.app.url as String to: https://{request.url.host}:8443\${request.url.path}
3. Open the **#oauth manager config** policy found in the Customizations/oauth manager folder. By default, no context variables are set.
4. Paste the copied assertion into the **#oauth manager config** policy.
5. Double-click the assertion, edit the port number, and click **OK**.
6. Save and Activate the #oauth manager config policy.

- Comment: Target Configuration Policy: "oauth manager config"
- Comment: === Set custom values for Context Variables below ===
- Comment: === Add any new Context Variables or extensions below ===
- Set Context Variable this.app.url as String to: <https://{request.url.host}:9443{request.url.path}>

OTK Authorization Server Configuration

This policy contains two references to the default 8443 port.

To set an alternate port number:

1. In Policy Manager, open the read-only **OTK Authorization Server Configuration** policy.
2. Copy the following Set Context Variable assertions:
 - host_login_server
 - host_content_server
3. Open the **#OTK Authorization Server Configuration** policy found in the Customizations/Authorization Server folder. By default, no context variables are set.
4. Paste the copied assertions into the **#OTK Authorization Server Configuration** policy.
5. Double-click the assertion, edit the port number, and click **OK**.
6. Save and Activate the **#OTK Authorization Server Configuration** policy.

- Comment: Target Configuration Policy: "OTK Authorization Server"
- Comment: === Set custom values for Context Variables below ===
- Comment: === Add any new Context Variables or extensions below ===
- Set Context Variable host_login_server as String to: <https://{gateway.cluster.hostname}:9443>
- Set Context Variable host_content_server as String to: <https://{gateway.cluster.hostname}:9443>

OTK Variable Configuration

This policy contains two references to the default 8443 port.

To set an alternate port number:

1. In Policy Manager, open the read-only **OTK Variable Configuration** policy. Expand the assertions.
2. Copy the following Set Context Variable assertions:
 - host_oauth2_auth_server
 - oauth2_server_port
3. Open the **#OTK Variable Configuration** policy found in the Customizations/Tools folder. This #policy may already contain custom settings for your oauth2 server hostname and certificate.
4. Paste the copied assertions into the **#OTK Variable Configuration** policy.
5. Double-click the assertions, edit the port number, and click **OK**.
6. Save and Activate the **#OTK Variable Configuration** policy.

- Comment: Target Configuration Policy: "OTK Variable Configuration"
- Comment: === Set custom values for Context Variables below ===
- Comment: === Add any new Context Variables or extensions below ===
- Set Context Variable oauth2_server_hostname as String to: myhost.com
- Set Context Variable oauth2_server_certificate as String to: myhost.com
- Set Context Variable host_oauth2_auth_server as String to: <https://{gateway.cluster.hostname}:9443>
- Set Context Variable oauth2_server_port as String to: [9443](#)

Configuring APIs

The APIs referencing the default 8443 can be configured directly.

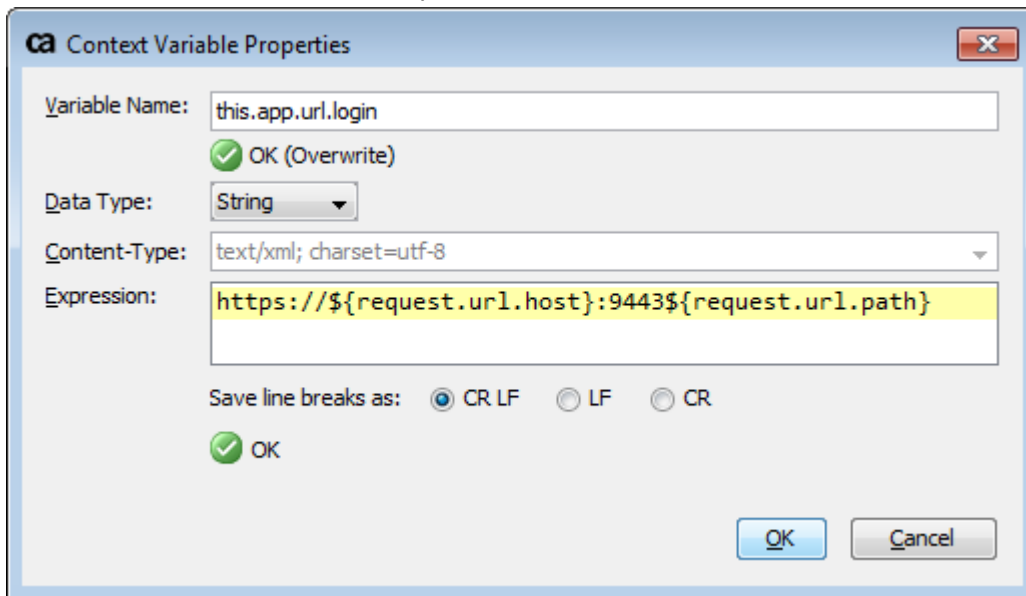
The following APIs both reference the default 8443 port for the **this.app.url.login** context variable:

- /oauth/manager/tokens
- /oauth/manager/clients

You perform the same procedure in both APIs.

To configure the API with an alternative port:

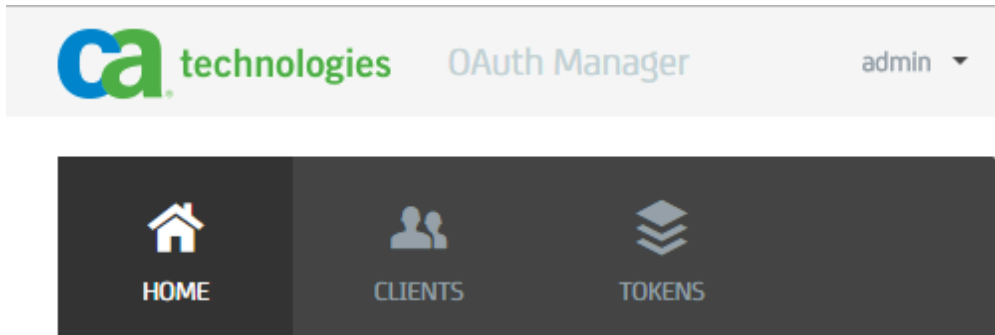
1. In Policy Manager, open the API.
2. Search for "8443" and locate the assertion that sets the **this.app.url.login** context variable.
3. Double click the assertion, edit the port number, and click **OK**.



4. **Save and Activate** the API.

Registering Clients with the OAuth Manager

The OAuth Manager displays information about registered OAuth clients and associated OAuth tokens that are used to access OAuth protected resources.



OAuth Manager

This management application is a simple tool to manage client applications and granted resource owner/client combinations

© CA Technologies. All rights reserved.

Perform the following tasks that are related to the OAuth Manager:

Log into the OAuth Manager

To log into OAuth Manager:

1. Open a browser and go to: `https://<yourgatewayURL>:8443/<instanceModifier>/oauth/manager`
2. Provide a username and password. The type of access you are granted depends on your user role. See [OTK User Role Configuration](#).
3. Click **Clients** to list, delete, and register OAuth clients.

Register a Client

To register a client:

1. Open the OAuth Manager.
2. Click **Clients**.
3. Click **Register a New Client**.

REGISTER A NEW CLIENT

4. Fill out the form using the fields below, then click **Register**:

Field	Field Option	Note
Client Name		Required. Do not use leading, trailing, or multiple spaces.
Organization		Required. Do not use leading, trailing, or multiple spaces.
Description		Optional
Registered By		Set to your logged in username.
Client Type	confidential	Default. More secure.
	public	Less secure.
Client Key		System generated UUID. You can overwrite it.
Authentication Method	Client Secret (Basic)	Client authenticates with authorization server by including the Client ID and Client Secret as client credentials using the HTTP Basic authentication scheme. This option is default.
	Client Secret (Post)	Client authenticates with authorization server. It includes the Client ID and Client Secret as client credentials in the body of the request.
	Client Secret (JWT)	Client authenticates with authorization server using JWT signed with client secret.
	Private Key (JWT)	Client authenticates with authorization server using a JWT signed with a private key. Using this option, either jwks or jwks_uri must be defined to share the public key with the Authorization server. See Client Authentication for more details.
jwks		The value of client JSON Web Key Set
jwks_uri		The URL of the client JSON Web Key Set. Example: https://[Gateway]:[port]/[file].json
Client Secret		System generated UUID. Maximum 255 characters.
Master Key		Applies only to clients using the CA Mobile API Gateway (MAG). Check the box to make the client key a master key. This enables dynamic client id creation. See MasterKey .
Status	ENABLED	Allows the client app to be used immediately after registration.
	DISABLED	Suspends use of the client app.

Scope		Provide the default scope requests for the client. Maximum 4000 characters. For details, see Scope .
Callback URL		Provide values. For details, see Callback .
Environment		The Environment value is not validated by the OTK by default. Set the value to the client's associated platform such as iOS, Android, or web. You can enter the environment value to filter search results from the List Keys page. You can also customize OTK policies to take advantage of environment information during OAuth related requests.

You have registered a new client.

Set the Master Key for Mobile Clients

The **Master Key** setting applies only to mobile clients using the CA Mobile API Gateway (MAG) and enables dynamic client id creation.

NOTE

If you are using MAG registered mobile clients, you must click Master Key to enable dynamic client id creation. Static client id creation is deprecated.

A master key allows the registered MAG client to retrieve new client credentials at the **/connect/client/initialize** default endpoint but not to consume any protected endpoints. The endpoint issues an original set of client credentials for an initial application registration, then upgrades client credentials for a known client_id. This dynamic client id creation allows for multiple instances of the same running app and is used for device-to-device login.

[mag] Master keys are used to initialize the client (e.g.: /connect/client/initialize)
Master Key

Click the check box to indicate that client key for this client is a master key. The option to specify a client secret is unavailable. When the key is added, the OAuth Manager automatically sets the client secret value to the client_id value. Similarly, if you manually set the client secret to the generated client key, the client is treated as having a master key.

Behavior is as follows:

- Only master keys can be used to access the /connect/client/initialize API which initializes the Mobile SDK with the MAG server. The API endpoint dynamically issues unique client credentials (non-master keys).
- Only clients with non-master keys can request OAuth tokens.
- If a master key is deleted, all keys that were issued based on that master key are also deleted. Any mobile app that is configured with a deleted key is disabled.
- For the first app on a non-registered device, a new set of client credentials is registered with the device-id.
- For subsequent apps, a new set of client credentials is registered with the username that is associated with the given device-identifier. Earlier client credentials that are associated with a device-id are updated to the username.

Set the Callback URL

The callback_uri is also known as the redirect_uri in OAuth 2.0.

For OAuth 2.0, the `redirect_uri` parameter may be used with response types.

The value of the `callback_uri` is a comma separated list of absolute URLs. You must include the scheme.

Valid callback_uri	Invalid callback_uri
<code>https://example.ca.com/callback,https://another-callback.ca.com/granted</code>	<code>example.ca.com</code>
<code>https://example.ca.com:9876/callback?key=value</code>	
<code>myscheme://for.my.mobile.native.app</code>	

Set Scope

The scope is a space-separated list of values that apply to OAuth 2.0 clients only and limit access for OAuth tokens. A client is initially registered with a set of scope values. These become the default scope requests for the client. For mobile clients, the default scope values are imported to the Mobile SDK through the `msso_config.json` file.

To restrict a scope request to a **subset** of the default set, set the scope request explicitly on the client side. A client cannot be granted a scope that does not belong to the registered default set. If the requested scope contains explicit scope requests that are outside of the registered default set, the access token is removed and a new access request is required using scopes within the default set.

To add a scope that is not in the default set, you must re-register the client with a new default set containing the new scope, and must export the `msso_config.json` file to the SDK.

An OAuth protected API can require specific scope values. Any `access_token` used at that API must be granted for all required scope values, otherwise the request fails. If no scope is registered and no scope is requested, scope is set to 'oob'.

Default supported scope values are as follows:

Open ID Connect Scope Values

Scope Value	Notes
<code>openid</code>	The OpenID Connect scope enables clients to send requests to the <code>/userinfo</code> endpoint. This scope causes the server to issue an <code>id_token</code> .
<code>address</code>	Requires <code>openid</code> scope.
<code>phone</code>	Requires <code>openid</code> scope.
<code>email</code>	Requires <code>openid</code> scope.
<code>profile</code>	Requires <code>openid</code> scope.
<code>user_role</code>	Requires <code>openid</code> scope. Returns the role of the <code>resource_owner</code> . By default it is user or admin . The role admin is used in the OAuth Manager and MAG Manager to identify an administrator. It has to be requested with 'openid'. This Scope value is an OTK extension; it is not part of OpenID Connect.

Mobile Single Sign-On Related Scope Values

The scope request determines the authorization that is granted to the client by the resource owner. If the requested scopes match or are a subset of the registered scope, access to the protected resource is granted. For mobile clients, the default scopes are imported to the Mobile SDK through the `msso_config.json` file.

Requires a Mobile API Gateway (MAG) installation.

Scope Value	Authentication Flow	Notes
<code>msso</code>	User Credentials	The username/password scope. Used for device registration with user credentials. Password grant type request. Requires <code>openid</code> scope. Clients requesting an <code>access_token</code> using the <code>msso</code> scope also receive an <code>id_token</code> that enables mobile single-sign on across multiple apps. Default endpoint for the user credentials workflow is <code>/connect/device/register</code> .
<code>msso_register</code>	User Credentials	The social login scope. Same as the <code>msso</code> scope, but is required for mobile single sign-on when using social login credentials. Follows the user credentials workflow for the password grant type. Default endpoint for the user credentials workflow is <code>/connect/device/register</code> .
<code>msso_client_register</code>	Client Credentials	The device to device scope. Required for mobile single sign-on with client credentials, not user credentials. Clients must be of type "confidential" to use the client credentials workflow. By default, clients using this scope can register a device. The client id and client secret are used for device registration. Default endpoint for the client credentials endpoint is <code>/connect/device/register/client</code> . This default endpoint is set in the MAG Variable Configuration assertion as the <code>device_register_client_endpoint_path</code> .

Mobile App Services Scope Values

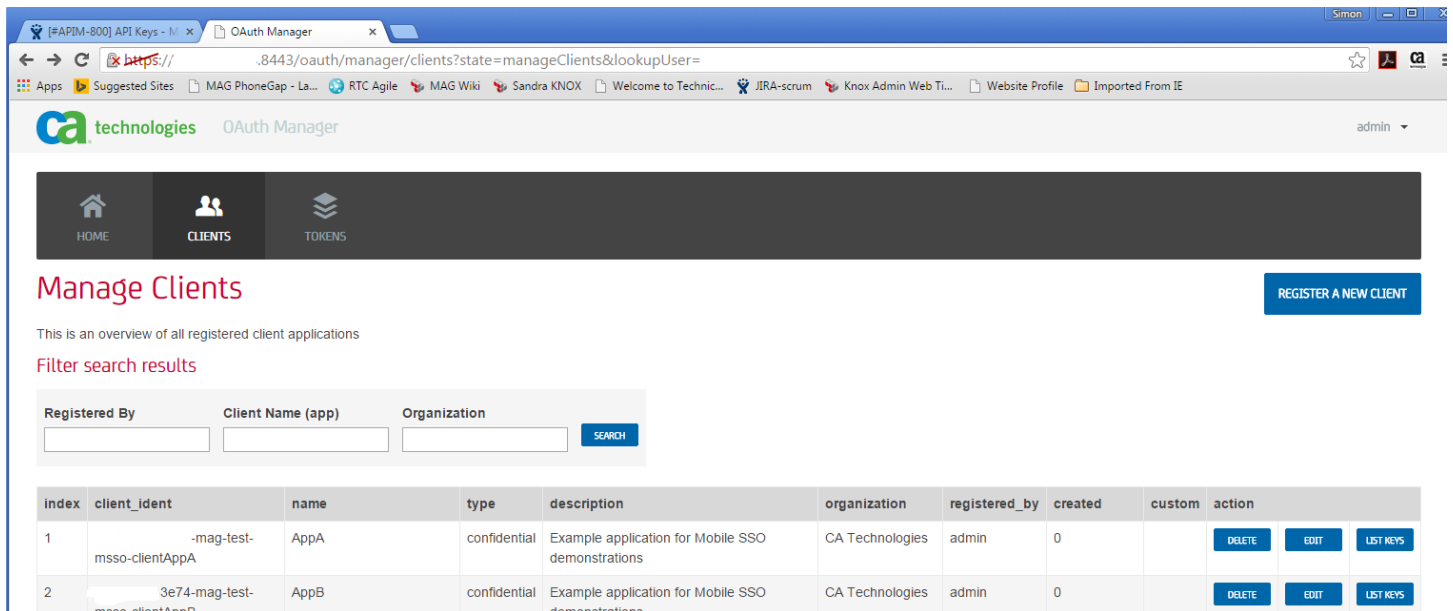
Requires a Mobile App Services installation.

MAS Service	Related Scopes	Notes
MAS Storage	<code>mas_storage</code>	Required to access MAS Storage functionality.. Used to Manage user and group access to messages from enterprise identity providers.
MAS Messaging	<code>mas_messaging</code>	Required to access MAS Messaging functionality. Supports messaging using the MQTT Pub/Sub model.

MAS Identity	mas_identity_retrieve_<resource> mas_identity_create_<resource> mas_identity_update_<resource> mas_identity_delete_<resource>	Used for clients to store user messages locally on the device or in the cloud. At least one of the scopes is required. Scopes permit the app to retrieve, create, update, and delete data for a resource. For example, mas_identity_retrieve_users allows the app to retrieve users. The specified <resource> value is one of the following SCIM resource types: <ul style="list-style-type: none"> • users • groups The following additional SCIM resource types are accessible by default when any of the valid mas_identity* scopes are requested: <ul style="list-style-type: none"> • ServiceProviderConfig • ResourceTypes • Schemas
--------------	--	---

Manage Clients

In the OTK, to request OAuth tokens and consume OAuth protected APIs, clients must be registered. The Manage Clients page allows you to add, edit, or delete clients.



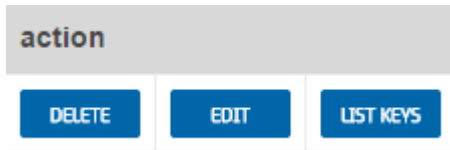
The screenshot shows the OAuth Manager interface. At the top, there are navigation tabs for HOME, CLIENTS, and TOKENS. The main heading is 'Manage Clients' with a 'REGISTER A NEW CLIENT' button. Below this, a search filter is provided with input fields for 'Registered By', 'Client Name (app)', and 'Organization', along with a 'SEARCH' button. The search results are displayed in a table with the following data:

index	client_id	name	type	description	organization	registered_by	created	custom	action
1	-mag-test-mss-clientAppA	AppA	confidential	Example application for Mobile SSO demonstrations	CA Technologies	admin	0		DELETE EDIT LIST KEYS
2	3e74-mag-test-mss-clientAppR	AppB	confidential	Example application for Mobile SSO demonstrations	CA Technologies	admin	0		DELETE EDIT LIST KEYS

NOTE

Client management is not available via the OAuth Manager if the CA OAuth Toolkit has been integrated with the CA API Portal.

Available Actions



The available actions are:

- **Delete** – If you delete the client application, all client IDs that are issued for the client are also deleted.
- **Edit** – Edit the client name, organization, description, and client type. Also allows you to set a value in JSON format for the client_custom field. Click **Update Client** to save any changes.
- **List Keys** – Displays individual client information. The following table shows additional information for selected fields.

List Client Keys

All registered client_keys for the given client application appear on the list client keys page.

Click **Add Client Key** to create additional keys for this client.

Available Actions



Perform any of the following available actions for a specific client key:

- **Revoke** – Deletes the client key. All tokens for this client key are also revoked.
- **Edit** – Edit properties for the key such as changing the status, disabling the key, adding scopes, and providing a Callback URL. Disabling a client key prevents the client key from being used for any future tokens, however it does not disable the tokens for that client key. Allows you to set a value in JSON format for the client_key_custom field. Allows you to set the Service IDs and Account Plan Mapping IDs values that are used in the CA API Portal.
- **Disable Tokens** – Disables all tokens for the client key.
- **Export** – Used to export client information in JSON message format. Accesses the current server configuration values. Use the JSON message to initialize OAuth clients or the CA Mobile API Gateway SDK for mobile applications.

Field Information

Field	Notes
client_ident	The identifier of the client.
client_key	In OAuth 2.0 the client_key = the client_id . They are synonymous terms. Maximum 255 characters.
secret	The client secret. If the client key and the secret are the same value, the client key is used as the master key in other endpoints/policies. Maximum 255 characters.

scope	The allowed scopes for the client. Maximum 4000 characters.
environment	Identifies the client platform.
callback	In OAuth 2.0, call back is the <code>redirect_uri</code> . Multiple URIs are supported.
expiration	The date until this key is valid. A value of 0 indicates that the key never expires.
status	Either "ENABLED" or "DISABLED". Disabled tokens cause resource requests to be denied.
client_key_custom	A custom field that is associated with the client key. The custom value must be a valid JSON object and cannot contain the following characters: < > &
serviceIds	A comma separated list of key strings that identify API services that are registered with the CA API Portal.
accountPlanMappingId	A comma separated list of key strings that identify account plans registered with the CA API Portal.

Manage Tokens

To manage tokens using the OAuth Manager:

1. Open a browser and navigate to this URL:
`https://<Gateway_host>:8443/oauth/manager`
The home page of the OAuth Manager appears.
2. Click **Tokens** to view values of issued tokens, and to disable or revoke tokens.
The following table shows additional information for selected fields.

Field	Description
rtoken	A <i>refresh_token</i> if available
reexpiration	The expiration date of the refresh token
client_key	In OAuth 2.0 it is the "client_id"
status	Disabled tokens cause resource requests to be denied

Available Actions



Perform any of the following available actions for a specific token:

- **Revoke** – Deletes the token.
- **Disable/Enable** – Toggles the current state of the token between invalid and valid.

Manage OAuth Clients with CA API Portal

NOTE

The following information applies only to the on-premise (classic) version of the CA API Developer Portal. No modification is required for the SaaS API Portal.

Integrating with the CA API Portal allows you to manage OAuth clients through the CA API Portal instead of through the OAuth Manager.

To manage OAuth clients with the CA API Portal:

- Meet the preconditions described in [Before You Begin](#)

After integration, OAuth clients do not appear in the OAuth Manager. Clients are identified by API key (OAuth client_id) and managed solely through the CA API Portal. However, you can still manage tokens through OAuth Manager.

Before You Begin

The following conditions must exist:

- /portalman/* services are deployed on the target gateway
- /api/keys/* services are deployed on the target gateway
- The assertion "Look Up API Key" is installed. The "Look Up API Key" assertion becomes available when you install CA API Portal.

Replace the OTK Client DB Get Extension

Replace the default **OTK client DB Get Extension** policy with a new policy available for download from the following instructions. The replacement policy sets the `{OTKDB}` variable to false and provides code that looks up client_id's from the API Portal rather than the OTK database. The extension targets the read-only **OTK Client DB Get** policy located in Policy Fragments/persistence/client.

To replace the OTK Client DB Get Extension:

1. Right-click the following link, select **Save link as...**, then save the OTK-4_ClassicPortal.xml file.
/content/dam/broadcom/techdocs/us/en/assets/docops/apimt/otk-4_classicportal.xml
2. In the Policy Manager, open the **OTK Client DB GET Extension** policy.
The OTK Client DB Get Extension policy is located in OTK/Customizations/persistence.
3. Click **Import Policy** and select the `OTK-4_ClassicPortal.xml` file.
The default policy code is replaced.
4. Click **Save and Activate**.

Register OAuth Test Clients

Optional. The test clients verify the integration of the OTK with the Portal.

Register the OTK test clients with the API portal by sending an HTTP PUT request to the target gateway.

NOTE

The OAuth test clients are not managed applications of the API Portal. They do not appear listed as new applications.

Create the Request

1. Create an HTTP PUT request using: `https://<yourGateway>:8443/portalman/1/api/keys`.
2. Set the content-type to "text/xml; charset=UTF-8"
3. Add an authorization header:
Authorization: Basic base64(username:password)
4. Right-click the XML icon, select **Save link as...** and save the RequestMessage3301.xml file. Copy the contents and use it as the message body for the request.
`#unique_131`
5. Update the value of **CallbackUrl** in the XML request message as follows:
 - Replace **HOST** with the hostname of the target gateway.
 - If policies are installed with a URL instance modifier, update the path component accordingly.
`<l7:CallbackUrl>https://HOST:8443/mag/manager</l7:CallbackUrl>`
6. Use a tool such as SOAPUI or Fiddler to send the request.

The API keys are now registered with API Portal.

APIs and Assertions

The OAuth Toolkit uses the APIs provided by the different components. The APIs can be used by any other third-party client.

The following APIs are provided and are required.

All APIs support HTTP GET and HTTP POST (content-type: *application/x-www-form-urlencoded*). The requirement for SSL can be configured by customizing the policy.

Values within brackets followed by a "?" are optional parameters; for example: (¶meter=value)?

Swagger Documentation for OAuth Server APIs

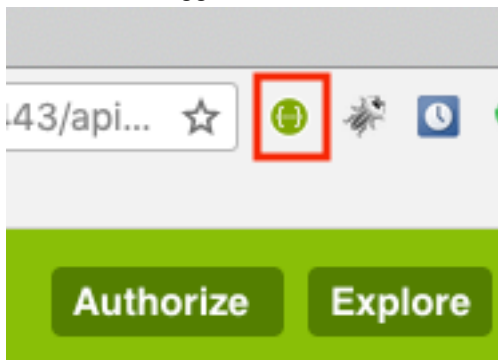
API definitions are available in Swagger for the OAuth Server APIs.

The swagger documentation presented on the docops.ca.com site is served from static JSON files rather than a running Gateway. To interact with the API through Swagger, access the `/apidocs/auth/oauth/v2/swagger` endpoint on your Gateway and view the documentation using a Swagger viewer such as Swagger UI.

Access API Documentation with the Swagger UI Chrome Extension

To use the Swagger UI Chrome Extension:

1. Install the Swagger UI chrome-extension.



2. In the browser URL field, enter the URL to point to the apidocs endpoint on your Gateway:
`https://<myGateway>:8443/apidocs/auth/oauth/v2/swagger`
The JSON file appears. This step is required to accept the certificate from the gateway.
3. Either enter the URL in Swagger UI field as shown below, or click the Swagger UI Console icon.



OTK Server APIs

All API's available in OAuth Toolkit server

Authorization Server APIs

Show/Hide | List Operations | Expand Operations

GET /auth/oauth/v2/authorize

GET /auth/oauth/v2/authorize/login

POST /auth/oauth/v2/authorize/login

OAuth Server API Endpoints

outputclass="supportedSubmitMethods">none

OAuth Toolkit APIs

The following APIs are available through the OAuth Toolkit:

none

OAuth Validation Point (OVP) API

There are several endpoints that are used to validate requests using OAuth.

This documentation explains the tasks associated with OVP. For the Swagger file, go to [OAuth Toolkit APIs](#).

NOTE

An endpoint containing "v2" means that it is used by OAuth 2.0. OAuth 1.0 support has ended.

Associated Tasks

Authorize a request_token

Authorizes a request_token, making it available in exchange for an access_token.

```
/oauth/validation/v1/authorize?token=<value>&expiration=<value>&verifier=<value>
```

token: the temporary token to be authorized

expiration: the new expiration date that will be used if the token is valid

verifier: the verifier used if the token is valid

Validation requirements:

- the expiration date has not expired
- a *resource_owner* must be assigned to the token
- a callback must be assigned to the token

Response:

- status: 200, content-type: text/xml, body
- status: 401, content-type: text/xml, body

Validate a refresh_token

```
/oauth/validation/validate/v2/refresh_token?client_key=<value>&rtoken= <value>&scope=<value>
```

client_key: the *client_key* issued for this token

rtoken: the *refresh_token* to be validated

scope: the scope to be used

Validation requirements:

- *client_key* must match the one that was used when the token was generated
- the expiration date must not be expired
- the status must be ENABLED

Response:

- status: 200, content-type: text/xml, body
- status: 401, content-type: text/xml, body

Validate OAuth Parameters and Signature

The OAuth Validation Point (OVP) is used when clients access resources. Validated tokens and signatures are cached to improve performance. Ensure the default values for caching conform to the security policy at your organization.

APIs	Notes
/oauth/validation/validate/v2/token	Used to validate "oauth" parameters. Validation depends on the token type. If token type is "MAC" or "BEARER", the expiration date and the status will be verified. The default <i>cacheAge</i> context variable is 60 seconds. A revoked token continues to be authorized for up to 60 seconds beyond revocation.
/oauth/validation/validate/v1/signature	Used to validate "oauth_signature" Validation requirements: <ul style="list-style-type: none"> • The signature is verified • The client_key is verified • expiration date has not expired • status is ENABLED • The token is verified • Expiration date has not expired • status is ENABLED (for access_tokens only) • For authorized request token, the verifier must exist in the tokenstore The default <i>cacheAge</i> context variable is 30 seconds. An accepted signature is cached for 30 seconds.

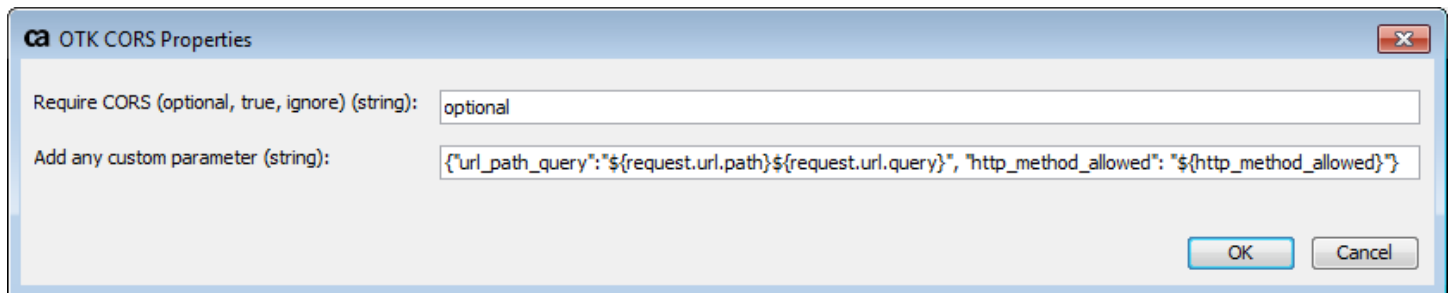
CORS Support for OTK APIs

Cross-Origin Resource Sharing (CORS) is a [W3C specification](#) that allows an API endpoint to accept cross-domain requests.

OTK CORS Default Behavior

The OTK CORS encapsulated assertion determines the following default behavior for CORS:

- CORS support is set to optional, which allows the client to determine whether CORS is supported. CORS is neither required, nor ignored.
- Preflight is set to false.
- Accepted Origins is set to accept all domains. (Access-Control-Allow-Origin: *)
- Allowed methods are GET, PUT, POST, DELETE, OPTIONS (Access-Control-Allow-Methods).



The OTK CORS encapsulated assertion applies to the following OAuth protected APIs:

- /auth/oauth/v2/token
- /connect/session/logout
- /apidocs/auth/oauth/v2/*
- /oauth/v2/protectedapi*
- /connect/session/status
- /openid/connect/v1/userinfo
- /auth/oauth/v2/client/export
- /auth/oauth/v2/token
- /auth/oauth/v2/token/revoke

Customize the Requirement of CORS for all OAuth Protected APIs

The following customization still uses the default CORS logic, but modifies the requirement of CORS across all APIs. It overrides the setting of requireCORS for each API.

To customize the requirement of CORS:

1. Open the #OTK CORS policy found in `OTK-version/customizations/cors`.
2. In the top left panel of the Policy Manager, with the Assertions tab selected, search for Set Context Variable.
3. Click this assertion and drag it into the #OTK CORS policy.
A Context Variable Properties dialog appears.
4. For **Variable Name** type: requireCORS
5. For **Expression** type one of the following values:
 - **True** – The API can only be used with CORS.
 - **Optional** – (Default) The client determines whether CORS is used or not.
 - **Ignore** – The API cannot be used with CORS. CORS request headers are ignored.

6. Click **OK**.
7. Leave the useDefaultCORS variable set to true.
8. **Save and Activate**.

Context Variable	Value
RequireCors	One of the following: <ul style="list-style-type: none"> • True – The API can only be used with CORS. • Optional – (Default) The client determines whether CORS is used or not. • Ignore – The API cannot be used with CORS. CORS request headers are ignored.

Customize the Requirement of CORS for a Specific API

Use the OTK CORS Extension policy to modify the OTK CORS encapsulated assertion per API. Modification takes the form of a JSON message.

Find the assertion in *OTK-version/Policy Fragments/configuration/cors*.

To customize the requirement of CORS for a specific API:

1. Open the #OTK CORS policy found in *OTK-version/customizations/cors*.
2. Double-click the "Set Context Variable useDefaultCors" assertion and set the value to `false`.
3. Open the OTK CORS extension policy found in *OTK-version/customizations/cors*.
4. Provide custom policy to override the default CORS behavior.

Error Code

Error code 134 indicates a CORS-related issue.

For example:

```
x-ca-err: 3003134
```

```
{ "error": "invalid_request", "error_description": "The request did not match CORS requirements" }
```

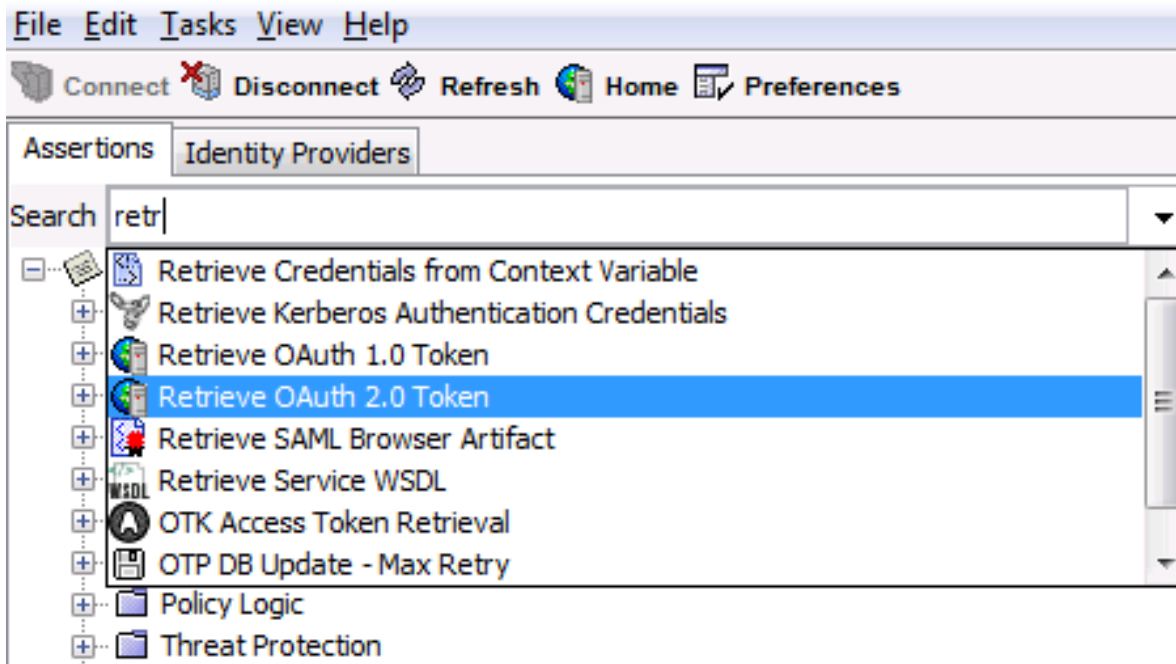
OAuth Client Assertions

NOTE

The OAuth Client Assertions described on this page are available only when the CA Mobile API Gateway is installed.

Use the following OAuth Client assertions to configure the Gateway as an OAuth client to consume OAuth-protected resources.

Access the OAuth Client assertions in the Policy Assertions panel under the XML Security folder.



Retrieve OAuth 1.0 Token Assertion

This assertion implements the two-stage handshaking process of OAuth 1.0.

Given the following:

- The user has an account on a website service that stores photos. To access the website, the user provides user credentials.
- The consumer is a third party application that wants to obtain access to the protected resource (photos) on a user's behalf.
- The user does not share the credentials to access the protected resource with the consumer.
- The consumer application is registered with the protected resource. The protected resource identifies the application by generating a consumer key and consumer secret.

In stage one, the consumer application requests access to the photo site. The request is directed to an authentication server and includes the unique consumer key identifying the application. The authentication server grants a temporary **OAuth request token**. The consumer application redirects the user to a callback URL on the photo site containing an **OAuth verifier**, where the user can explicitly allow (authorizes) the consumer application access to the photos.

In the second stage, the consumer application requests an **OAuth access token**, using the consumer key, the request token, and the verifier. The photo website grants the access token. The consumer application uses the access token to access the protected resource.

Context Variables Set

oauth.access_token
oauth.access_token_secret
oauth.auth_req_url
oauth.full_access_token_response

oauth.full_oauth_token_response
oauth.oauth_token
oauth.oauth_token_secret

Context Variables Used

oauth_callback
oauth_token
oauth_token_secret
oauth_verifier
request.http.method

Consume OAuth 1.0 Resource

The Consume OAuth 1.0 Resource assertion creates the OAuth authorization header. The header can be added to the HTTP Authorization header to consume OAuth 1.0 protected resources.

Context Variables Set

oauth.resource_authorization_header

Context Variables Used

access_token
access_token_secret
request.http.method

Retrieve OAuth 2.0 Token Assertion

The Retrieve OAuth 2.0 Token assertion is used to retrieve an access token from the authorization server. The token response from the authorization server may include a refresh token.

For authorization code and implicit grant types, this assertion implements the two-stage handshaking process of OAuth 2.0. It first returns the authorization request URL in a context variable. The client can then redirect the user-agent to the authorization request URL. After the authorization server authenticates the resource owner and access is granted, this assertion will be called back to perform the second stage handshaking, which will then return access token.

Context Variables Set

oauth.access_token
oauth.auth_req_url
oauth.full_token
oauth.refresh_token

Refresh OAuth 2.0 Token Assertion

The Refresh OAuth 2.0 Token assertion is used to refresh an access token. An access token can be refreshed only if the authorization server issued a refresh token.

Context Variables Set

oauth.access_token

oauth.full_token


oauth.refresh_token

Encapsulated Assertions

To find OTK related encapsulated assertions:

1. In the Policy Manager, go to **Tasks, Extensions and Add-Ons, Manage Encapsulated Assertions**.
2. Type OTK to filter the list. All OAuth encapsulated assertions start with OTK.
3. Select the encapsulated assertion and click **Properties** to view details.

Encapsulated Assertion Configuration Properties

Name and Icon:  OTK Require OAuth 2.0 Token

Palette Folder: Internal Assertions

Policy: OTK Require OAuth 2.0 Token Set Policy

Description: Validates an OAuth 2.0 access_token for a request. By default the request will fail if the given access_token has expired or does not match the given configuration.

Inputs:

GUI	Name	Type	Label
<input checked="" type="checkbox"/>	scope_required	String	Required SCOPE(s)
<input checked="" type="checkbox"/>	cache_lifetime	Integer	Cache validation result (s)
<input checked="" type="checkbox"/>	onetime	Boolean	Is this a one-time access_tok...
<input checked="" type="checkbox"/>	scope_fail	Boolean	Fail if this SCOPE was granted?
<input checked="" type="checkbox"/>	given_access_token	String	Access Token (Bearer only, o...

Outputs:

Name	Type
access_token	String
content-type	String
error.code	String

Update routing statistics in parent policy Allow debug tracing into backing policy

OK Cancel

In the following descriptions, the term "encas" is used as an abbreviation of "encapsulated assertion".

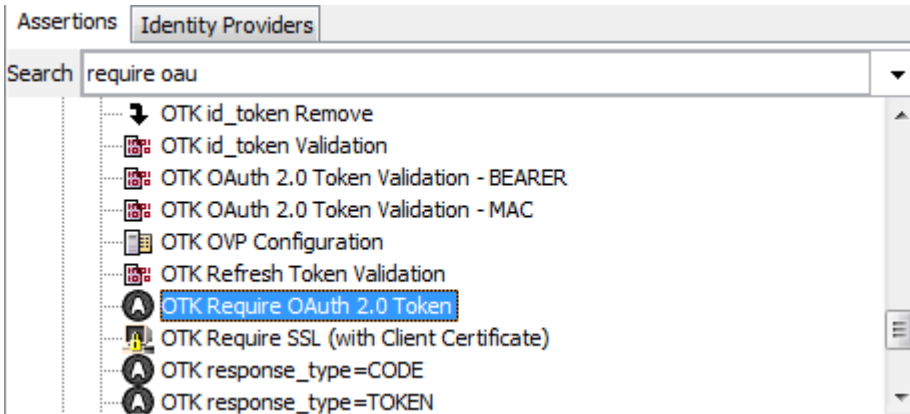
Key encapsulated assertions include:

NOTE

This is not a comprehensive list of all encapsulated assertions. All other undocumented OTK encapsulated assertions can be used as it is.

OTK Require OAuth 2.0 Token

Use this encapsulated assertion to allow access only when a valid access_token is presented by the client. The assertion searches for access_tokens presented in the authorization header as a query parameter or as a post body parameter. Use this assertion as early as possible in an API policy.



This assertion uses the OTK Access Token Retrieval assertion to find the incoming `access_token`. Refer to that description for error messages.

Drag the assertion into a policy and configure the properties shown in the table below.

Properties	Parameter Name	Type	Notes
Required SCOPE(s)	<code>scope_required</code>	String	If SCOPE is not required, this value can be empty. A space separated list of required SCOPEs. An <code>access_token</code> is only accepted if it has been granted with those SCOPE values.
Cache validation result (s)	<code>cache_lifetime</code>	Integer	This value cannot be empty. Represents the time in seconds for which an <code>access_token</code> is cached. The assertion initially validates an <code>access_token</code> . The validation result is then cached until the cache period expires. This increases performance, but also enables clients to use potentially expired <code>access_tokens</code> . The <code>cache_lifetime</code> value extends the lifetime of the token. A value of 0 indicates no caching is performed.
Is this a one-time access-token?	<code>onetime</code>	Boolean	Default value: false . To allow an <code>access_token</code> to be considered valid only once for this endpoint, set this value to true . This setting is rare, but enables special use cases.
Fail if this SCOPE was granted?	<code>scope_fail</code>	Boolean	Default value: false . Set this value to "true" if a request should fail in the case that an <code>access_token</code> has been granted for at least one the specified SCOPE values listed above

Access Token	given_access_token	String	Optional. The hardcoded value of an access token or a context variable representing an access_token. Use this property if an access_token is made available, but not by the client, or if the access_token is passed using a non-standard mechanism.
--------------	--------------------	--------	--

Context Variables

The encas sets the following context variables:

Context Variable	Notes
`\${session.client_id}`	The client_id of the client that has received the token
`\${session.subscriber_id}`	the username of the resource_owner that has granted access for the client and therefore to access the endpoint
`\${session.scope}`	the SCOPE that was granted for this access_token
`\${session.expires_at}`	the expiration time in seconds
`\${access_token}`	the access_token that was used with this request

OTK Access Token Retrieval

This encas is used within OTK Require OAuth 2.0 Token where it extracts an access_token from a request. It works on given input variables and searches for an access_token. When the token is found, it is passed in as an authorization header value:

authorization: Bearer <token>, authorization: MAC <mac-values>

Input Field	Parameter Name	Type	Notes
Allow Authorization Header	allow_header	Boolean	Indicates whether to accept an access_token within the authorization header.
Allow Parameter	allow_query	Boolean	Indicates whether to accept and access_token as query or from post parameter.
Authentication Header	auth_header	String	The content of the authorization header to be used. Values with scheme "Bearer" and "MAC" are accepted
Parameter	auth_param_token	String	The content of the variable that contains the access_token to be used

Access Token	given_access_token	String	The hardcoded value of an access token or a context variable representing an access_token. Allows you to provide an access_token other than from a well known location. This is useful if the access_token is held somewhere but not passed in with a request.
--------------	--------------------	--------	--

Context Variables

The encas sets the following context variables:

Context Variable	Notes
\${access_token}	The client_id of the client that has received the token
\${auth_header}	The username of the resource_owner that has granted access to the client and therefore access to the endpoint
\${auth_scheme}	The token scheme. One of the following values: <ul style="list-style-type: none"> • Bearer • MAC

The assertion creates an error if no access token is found or if multiple locations within the request contained a token.

```

HTTP Header:
Status: 401
Content-type: application/json
Pragma: no-cache
Cache-Control: no-store
HTTP Body:
{
  "error": "invalid_request",
  "error_description": "Missing or duplicate token"
}

```

OTK Client Persist

The OTK model allows multiple OAuth clients and more than one client ID per client. The OTK Client Persist encas persists a new OAuth client. By default there is no need to use this encas directly. It is used within OAuth Manager to register a new client. Nevertheless, it may be useful for testing purposes or special cases.

Input Field	Parameter Name	Flag
Client Identifier	client_ident	Required.
Client ID	client_key	new client ID
Client Secret	client_secret	new client ID
Redirect URI	callback	new client ID optional
Description	description	new client
Environment	environment	new client ID optional

Expiration	expiration	new client ID
Client Name	client_name	new client
Organization	org	new client
Registered By	registered_by	Required.
Valid SCOPE	scope	new client ID optional
Status	client_status	new client ID
Client Type	type	new client
New Client	persist_client	Boolean
New Client ID	persist_client_key	Boolean
New Client including client ID	persist_client_and_key	Boolean

Create a New Client

To create a new client:

1. Set all required values.
2. Set all values flagged as "new client" (unless optional).
3. Set New Client boolean to **true**.
4. Ensure that the following values are unique:
 - Client Identifier
 - Client Name
 - Client Name with Organization

Create a New Client ID

To create a new client ID:

1. Set all required values.
2. The Client Identifier value must match an existing value.
3. Set all values flagged as "new client ID" (unless optional).
4. Set New Client ID to **true**.
5. Ensure that the Client ID value is unique.

Create a New Client with a New Client ID

To create a new client with a new client ID:

1. Set all values (unless optional).
2. Set New Client including client ID to **true**.

Context Variables

The encas sets the following context variables to the values shown if no error occurs:

Context Variable	Value
\${status}	200
\${result}	persisted
\${content_type}	text/plain

If an error occurs:

```
Status = 403
Content-type = application/json
Message:
{
  "error": "Client registration failed",
  "error_description": "The client could not be registered"
}
```

OTK SCOPE Verification

Use this encapsulated assertion whenever an API should process a request depending on the SCOPE that was granted to an `access_token`. Multiple instances of this encas can be used within the same policy.

The MAG policies use this encas at the endpoint `/opened/connect/v1/userinfo`. Depending on the granted SCOPE the endpoint returns an email address, profile information, and other values.

In a policy position this encas is always used after using "OTK Require OAuth 2.0 Token".

Input Field	Parameter	Notes
Granted SCOPE	<code>scope.granted</code>	Insert the values that were granted to the used <code>access_token</code> . In combination with "OTK Require OAuth 2.0 Token" simply use the example shown
Required SCOPE	<code>scope.required</code>	Insert the SCOPE value that must be found in the list of the granted SCOPE values. The encas will fail if the required SCOPE cannot be found.
Fail if SCOPE is found (true false)?	<code>fail</code>	Set this value to "true" if a request should fail in the case that an <code>access_token</code> has been granted for at least one the specified SCOPE values listed above.

Context Variables

No context variables are set by this encas.

Error Codes

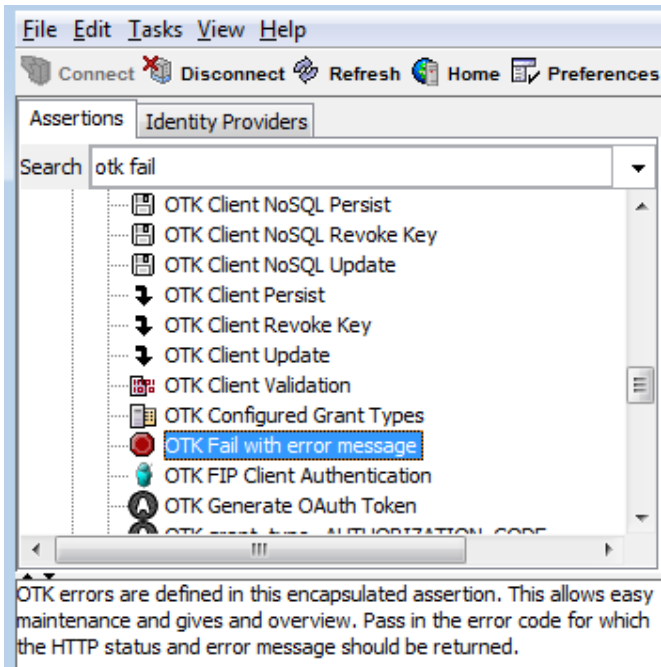
Error codes are structured as follows: 4-digits (identifies the API) + 3-digits (identifies the actual error).

Refer to the following sections:

How to Add Error Codes to a Policy

Error codes are structured as follows: 4-digits (identifies the API) + 3-digits (identifies the actual error).
















Drag the **OTK Fail with error message** encapsulated assertion into a policy to provide standardized error responses when error conditions are met. Using this single encapsulated assertion replaces having to define each error code explicitly in each policy.



For protected APIs, you can create your own codes by editing the **#OTK Fail with error message** policy. Use codes in the 700 to 899 range. Do not change or overwrite the existing error codes.

To add error handling to a policy:

1. In the Assertions pane, locate the **OTK Fail with error message** assertion.
2. Drag the assertion into your policy where you logically want to trigger the error. On release, the configuration dialog appears.
3. For **Error Code**, provide the 3-digit number that identifies the actual error. Use existing codes, or create custom codes in the 700 to 899 range. Refer to the Error Codes below for examples.
4. For **API prefix for error code**, provide a 4-digit number to identifier your API or use `${apiPrefix}` to reference a pre-defined API prefix.

- 2  Comment: Default assertions for a typical OAuth 2.0 protected API
- 3  Error Customize Error Response Override default gateway error response
- 4  POST GET OTK HTTP Method Validation Specify valid HTTP methods here
- 5  SSL OTK Require SSL (with Client Certificate) Require SSL, optionally require a client certificate
- 6  Optional OTK FIP Client Authentication Required if previous line was configured to require a client certificate!
- 7  OAuth token OTK Require OAuth 2.0 Token Searches for and validates an OAuth 2.0 access_token
- 8  Comment: === Implement API related validation here
- 9  initialize Set Context Variable myparam as String to: `${request.http.parameter.myparam}` Initialize variable here
- 10  parameters At least one assertion must evaluate to true check for required parameters
- 11  Compare Variable: `${myparam}` is not empty; If Multivalued all values must pass
- 12  error All assertions must evaluate to true Missing or empty required parameter
- 13  Add Audit Details: "Required parameter 'myparam' is empty or missing"
- 14  103 OTK Fail with error message Missing parameter
- 15  Comment: === Implement API related logic here
- 16  Comment: Backend call, transformations, ...

Commonly returned codes are:

1. xxxx201: The client authentication failed
2. xxxx202: The resource_owner authentication failed
3. xxxx203: SSL was required but not used by the client
4. xxxx204: SSL with client authentication was required but not used by the client
5. xxxx990: The token has expired
6. xxxx991: The access_token has not been granted for the required SCOPE
7. xxxx992: No access_token was included in the request, An access_token was included more than once
8. xxxx993: The token is disabled which means that associated client is disabled
9. xxxx000: Indicates that something unexpected went wrong

Error Handling

The Customize Error Response assertion (line 3) is required and sets up the error handling mechanism. By modifying this assertion, you can override the default error handling response. It precedes the OTK Require OAuth 2.0 Token assertion (line 7) in a policy. The complete list of error codes and messages are referenced in the OTK Fail with error message encapsulated assertion (line 14).

By default, error codes are returned in the HTTP response message header

```
x-ca-err
```

field. For example:

```
x-ca-err: 3007103
```

Additionally:

- `allow` is added to the response if the given HTTP method is not valid. It contains a comma-separated list of valid methods
- `www-authenticate` is added to the response if the given client or resource owner credentials are missing or could not be validated

The message body contains:

```
{ "error": "invalid_request", "error_description": "Missing or duplicate parameters" }
```

Error Codes (3-Digits)

The following table shows the 3-digit error codes for the OTK. This table has been updated for version 4.0.

Error code	HTTP status	Feature	Category	Info	Description
000	500	ALL	invalid_request	invalid	The request failed due some unknown reason
103	400	OTK	invalid_request	invalid parameters	Required parameters or headers were missing in the request
110	400	OTK	invalid_request	invalid session	The session has expired
111	429	OTK	invalid_request	Too many requests	The number of permitted requests has been exceeded
112	400	OTK	invalid_request	invalid PKCE request	The server is misconfigured so that the public key hash cannot be created
113	400	OTK	invalid_request	invalid grant	The given token (authorization code) is not valid
114	400	OTK	invalid_request	invalid redirect_uri	Due to an invalid or mismatching redirection URI.
115	400	OTK	invalid_scope	invalid scope was requested	No registered scope values valid for this client has been requested
116	400	OTK	invalid_response_type	unsupported response type	None of the supported response_types were used. Currently supported: 'code', 'token', 'token id_token' (in conjunction with scope=openid)
117	400	OTK	unauthorized_client	unauthorized client for this request	The client misses authorization for this request. By default this error does not occur but it may happen in certain custom environments

118	503	OTK	unsupported token type	unsupported token type	The server does not support the given type of token
119	400	OTK	unsupported_grant_type	the given grant_type is not supported	The given grant_type is not supported by the server
120	400	OTK	invalid_request	invalid id_token	The id_token may have expired or had missing claims
121	400	OTK	invalid_request	invalid JWT	The given JWT is not valid
122	400	OTK	invalid_request	invalid JWT/ id_token request	The JWT/ id_token could not be created
123	400	OTK	authentication_error	authentication was denied	The resource_owner cancelled authentication
124	400	OTK	authorization_error	authorization was denied	The resource_owner has denied access to resources.
125	400	OTK	invalid_request	invalid verifier	A verifier has already been set
126	400	OTK	invalid_request	invalid status	The client status is not as expected
127	400	OTK	invalid_request	expired client id	The given client id has expired
128	400	OTK	invalid_client	invalid client name	The given client name is not valid
129	400	OTK	invalid_request	invalid environment	The given environment is not valid
130	400	OTK	invalid_request	invalid client type	The client type is not valid
131	400	OTK	invalid_request	invalid id_token request	The id_token could not be created
132	400	OTK	invalid_request	invalid server configuration	The server has not been configured correctly
133	400	OTK	invalid_request	not a valid JSON object	The given message is not a valid JSON object
134	400	OTK	invalid_request	CORS requirements were not met	The request did not match CORS requirements
135	400	OTK	invalid_request	Token limit reached	The maximum number of tokens has been reached
136	400	OTK	invalid_configuration	Public Key Hash could not be created	The public key hash could not be created. Contact the administrator

201	401	OTK	invalid_request	invalid client	The given client credentials were not valid
202	401	OTK	invalid_request	invalid resource owner	The given resource owner credentials were not valid
203	403	OTK	invalid_request	missing SSL	The client did not use SSL
204	403	OTK	invalid_request	invalid request	Authentication failed
205	401	OTK	invalid_client	invalid client certificate	The given client credentials were not valid
300	400	OTK	invalid_request	client could not be persisted	The request contained values that are not unique on the server
301	400	OTK	invalid_request	client could not be persisted	The request used an invalid value for 'persist_type'
303	400	OTK	invalid_request	invalid operation	An invalid path was used
304	302	OTK	consent_required	consent is required	The authorization server requires user consent
305	400	OTK	invalid_request	invalid jwks and jwks_uri	Both jwks and jwks_uri are provided at OIDC registration. The jwks_uri and jwks parameters MUST NOT be used together.
306	401	OTK	invalid_client	invalid client	The given JWT for client authentication is not valid
307	401	OTK	invalid_client	invalid client	The request is a replay attack
308	401	OTK	invalid_client	invalid client	The replay attack protection could not be applied
990	401	OTK	invalid_request	invalid access_token	The access_token has expired
991	401	OTK	invalid_request	invalid access_token	The access_token lacks permissions (SCOPE)
992	401	OTK	invalid_request	invalid access_token	The access_token is missing or it has been provided more than once
993	401	OTK	invalid_request	invalid token	The token is disabled

API Identifier + Error Codes

The following error codes are defined in the OTK Fail with error message encapsulated assertion.

Error groups contain one or more APIs. To find an error, click the group then search for the specific error code.

NOTE

When an unexpected content_type is encountered, an error is returned, but the error code is inaccurate.

For error codes 1000-3000, see the CA Mobile API Gateway wiki; docops.ca.com/mag

This table has been updated for version 4.0.

Group/ API_ID	Code	Category	Info	Reasons	How to resolve
3000/ request_authorization_init	3000000	invalid_request	invalid		
	3000103	invalid_request	invalid parameters	<ul style="list-style-type: none"> - The request did not include headers and/or parameters as specified for the API - The request included duplicate parameters - The request included required headers or parameters but without value, empty 	Repeat the request including all required parameters and/or headers
	3000107	invalid_request	invalid mag-identifier	- The mag-identifier is not associated with a device	Repeat the request using a valid mag-identifier
	3000108	invalid_request	invalid mag-identifier	- The referenced device is in status 'registered' and has to be changed to 'activated' before this request can succeed	Either use MAG Manager to change the device status or contact the system administrator to do so
	3000112	invalid_request	invalid PKCE request	<ul style="list-style-type: none"> - the given code_challenge is invalid - the given code_challenge_method is not supported 	Repeat the request using a valid code_challenge or supported code_challenge_method

	3000114	invalid_request	invalid redirect_uri	<ul style="list-style-type: none"> - None of the registered redirect_uri's were used - no redirect_uri given: open redirect_uri's are not supported and therefore the redirect_uri has to be provided - no redirect_uri given: if the client is of type 'public' a registered redirect_uri has to be provided - no redirect_uri given: if multiple redirect_uri's are registered one has to be passed in - The format is not valid, e.g.: it does not have a scheme (https://, myscheme://) 	Repeat the request using a valid redirect_uri
	3000115	invalid_scope	invalid scope was requested	<ul style="list-style-type: none"> - No scope value matched a registered one for this client - Multiple scopes were requested but not separated by a space (' ') character 	Repeat the request using valid scope values
	3000116	invalid_response_type	unsupported response type	<ul style="list-style-type: none"> - The response_type was malformed or is unknown - The response_type 'token id_token' was requested but the requested scope did not include 'openid' - The response_type 'token id_token' was requested but the request did not include a 'nonce' 	Repeat the request using a valid response_type

	3000117	unauthorized_client	unauthorized client for this request	<ul style="list-style-type: none"> - The client may not be valid for the used response_type - The client may not be valid due to the 'type' - A confidential client using an implicit grant must have a registered redirect_uri - A public client must have a registered redirect_uri 	Verify the configuration of the client and repeat the request.
	3000130	invalid_request	invalid client type	<ul style="list-style-type: none"> - The client is of type public but has no registered redirect_uri - The confidential client is using an implicit grant type but has no registered redirect_uri 	Check if the client type for the client is configured as expected
	3000201	invalid_request	invalid client	<ul style="list-style-type: none"> - The given client_id or client_secret is not valid - The client_id and/or client_secret is missing - The credentials were not provided as base64 encoded value - The given client_id is configured as 'Master Key' - The client_id is disabled - The client_id has expired (if it had a limited lifetime) 	Repeat the request using valid credentials. If the error still occurs contact the system administrator
	3000203	invalid_request	missing SSL	<ul style="list-style-type: none"> - The client did not use 'https' but 'http' 	Repeat the request using 'https'
3001/ request_authorization_login	3001000	invalid_request	invalid		

	3001103	invalid_request	invalid parameters	<ul style="list-style-type: none"> - The request did not include headers and/or parameters as specified for the API - The request included duplicate parameters - The request included required headers or parameters but without value, empty 	Repeat the request including all required parameters and/or headers
	3001110	invalid_request	invalid session	<ul style="list-style-type: none"> - The session's lifetime has expired - The session has been granted and therefore used 	Repeat the authorization request and have the resource_owner authenticate before the session times out
	3001114	invalid_request	invalid redirect_uri	<ul style="list-style-type: none"> - None of the registered redirect_uri's were used - For missing redirect_uri: Open redirect_uri's are not supported and therefore the the redirect_uri has to be provided - For missing redirect_uri: If the client is of type 'public' a registered redirect_uri has to be provided - The format is not valid, e.g.: two redirect_uri's were included 	Repeat the request using a valid redirect_uri
	3001123	authentication_error	authentication was denied	<ul style="list-style-type: none"> - The resource_owner has denied login. 'action' was set to 'cancel' instead of 'login' 	Convince the resource_owner to login
	3001202	invalid_request	invalid resource owner	<ul style="list-style-type: none"> - The given username or password is not valid - The username and/or password is missing - The social login credentials were invalid - The cookie has expired 	Repeat the request using valid credentials. If the error still occurs contact the system administrator

	3001203	invalid_request	missing SSL	- The client did not use 'https' but 'http'	Repeat the request using 'https'
3002/ request_authorization_consent	3002000	invalid_request	invalid		
	3002103	invalid_request	invalid parameters	- The request did not include headers and/or parameters as specified for the API - The request included duplicate parameters - The request included required headers or parameters but without value, empty	Repeat the request including all required parameters and/or headers
	3002110	invalid_request	invalid session	- Access to resources was granted but after the session timed out - Access to resources was denied but after the session timed out - The session has been granted and therefore used	Repeat the authorization process and have the resource_owner authenticate before the session times out
	3002124	authorization_error	authorization was denied	- The resource_owner has denied access to resources. 'action' was set to 'Denied' instead of 'Grant'	Convince the resource_owner to grant access
	3002135	invalid_request	Token limit reached	- The maximum number of access tokens has been reached for the given Client (App) and Resource Owner	Revoke any unused tokens or increase the max number of allowed access tokens
	3002203	invalid_request	missing SSL	- The client did not use 'https' but 'http'	Repeat the request using 'https'
3003/ request_token_password_flow, request_token_code_flow, request_token_refresh_flow, request_token_client_creds_flow, request_token_jwt_flow, request_token_saml_flow	3003000	invalid_request	invalid		

	3003103	invalid_request	invalid parameters	<ul style="list-style-type: none"> - The request did not include headers and/or parameters as specified for the API - The request included duplicate parameters - The request included required headers or parameters but without value, empty 	Repeat the request including all required parameters and/or headers
	3003107	invalid_request	invalid mag-identifier	<ul style="list-style-type: none"> - The mag-identifier is not associated with a device 	Repeat the request using a valid mag-identifier
	3003113	invalid_request	invalid grant	<ul style="list-style-type: none"> - The given authorization code has expired - The given authorization code has been used already - The given authorization code is invalid - The given refresh_token has expired - The given refresh_token has been revoked 	Repeat the authorization process using a valid grant
	3003114	invalid_request	invalid redirect_uri	<ul style="list-style-type: none"> - The code to was requested using a different redirect_uri 	Repeat the request using a valid redirect_uri
	3003115	invalid_scope	invalid scope was requested	<ul style="list-style-type: none"> - No scope value matched a registered one for this client - Multiple scopes were requested but not separated by a space (' ') character - The refresh_token request included other or more SCOPE values than initially issued 	Repeat the request using valid scope values
	3003117	unauthorized_client	unauthorized client for this request	<ul style="list-style-type: none"> - The client was not the recipient of the given token 	Use a valid client; the one that initially received the refresh_token
	3003119	unsupported_grant_type	the given grant_type is not supported	<ul style="list-style-type: none"> - and invalid grant_type was used 	Repeat the request using a valid grant_type

	3003134	invalid_request	CORS requirements were not met	- The request did not provide CORS-required request headers	Verify that the user-agent (browser) supports CORS and adds required headers to the request
	3003201	invalid_request	invalid client	<ul style="list-style-type: none"> - The given client_id or client_secret is not valid - The client_id and/or client_secret is missing - The credentials were not provided as base64 encoded value - The given client_id is configured as 'Master Key' - The client is disabled - The client_id has expired (if it had a limited lifetime) 	Repeat the request using valid credentials. If the error still occurs, contact the system administrator
	3003202	invalid_request	invalid resource owner	<ul style="list-style-type: none"> - The given username or password is not valid - The username and/or password is missing - The JWT (id_token) has expired - The JWT (id_token) has an invalid signature - The JWT (id_token) is unknown. Only JWT that were issued by the server can be used - The JWT (id_token) was revoked - The SAML token has expired - The SAML token has has an invalid signature 	Repeat the request using valid credentials. If the error still occurs, contact the system administrator
	3003203	invalid_request	missing SSL	- The client did not use 'https' but 'http'	Repeat the request using 'https'
	3003993	invalid_request	invalid token	- The refresh_token is disabled which means that associated client is disabled	The token (client) has to be enabled or revoked and a new one requested.

3004/ revoke_token	3004000	invalid_request	invalid		
	3004103	invalid_request	invalid parameters	<ul style="list-style-type: none"> - The request did not include headers and/or parameters as specified for the API - The request included duplicate parameters - The request included required headers or parameters but without value, empty 	Repeat the request including all required parameters and/or headers
	3004117	unauthorized_client	unauthorized client for this request	<ul style="list-style-type: none"> - The client was not the recipient of the given token 	Use a valid client
	3004118	unsupported token type	unsupported token type	<ul style="list-style-type: none"> - The given token is of a type that is not supported - Neither a Bearer 'access_token' nor 'refresh_token' was used 	Contact the system administrator
	3004134	invalid_request	CORS requirements were not met	<ul style="list-style-type: none"> - The request did not provide CORS-required request headers 	Verify that the user-agent (browser) supports CORS and adds required headers to the request
	3004201	invalid_request	invalid client	<ul style="list-style-type: none"> - The given client_id or client_secret is not valid - The client_id and/or client_secret is missing - The credentials were not provided as base64 encoded value - The client is disabled - The client_id has expired (if it had a limited lifetime) 	Repeat the request using valid credentials. If the error still occurs contact the system administrator
	3004203	invalid_request	missing SSL	<ul style="list-style-type: none"> - The client did not use 'https' but 'http' 	Repeat the request using 'https'
3005/ client_details_export	3005000	invalid_request	invalid		

	3005103	invalid_request	invalid parameters	<ul style="list-style-type: none"> - The request did not include headers and/or parameters as specified for the API - The request included duplicate parameters - The request included required headers or parameters but without value, empty 	Repeat the request including all required parameters and/or headers
	3005132	invalid_request	invalid server configuration	<ul style="list-style-type: none"> - The server is searching for an unknown certificate 	Verify that the server has been configured to search for its public SSL cert via the correct certificate alias
	3005134	invalid_request	CORS requirements were not met	<ul style="list-style-type: none"> - The request did not provide CORS required request headers 	Verify that the user-agent (browser) supports CORS and adds required headers to the request
	3005201	invalid_request	invalid client	<ul style="list-style-type: none"> - The given client_id is unknown - The client is disabled 	Check with OAuth Manager for verify the current state of the client
	3005202	invalid_request	invalid resource owner	<ul style="list-style-type: none"> - The given username or password is not valid - The username and/or password is missing 	Repeat the request using valid credentials. If the error still occurs contact the system administrator
	3005203	invalid_request	missing SSL	<ul style="list-style-type: none"> - The client did not use 'https' but 'http' 	Repeat the request using 'https'
3006/ resource_owner_logout	3006000	invalid_request	invalid		
	3006103	invalid_request	invalid parameters	<ul style="list-style-type: none"> - The request did not include headers and/or parameters as specified for the API - The request included duplicate parameters - The request included required headers or parameters but without value, empty 	Repeat the request including all required parameters and/or headers

	3006107	invalid_request	invalid mag-identifier	<ul style="list-style-type: none"> - The mag-identifier is not associated with a device - The device was registered as a mobile client (SCOPE=mssso) but no mag-identifier was included 	Repeat the request including a valid mag-identifier
	3006117	unauthorized_client	unauthorized client for this request	<ul style="list-style-type: none"> - The client is not a valid user of this token 	Use a valid client whose client_id is found within the 'azp' key of the id_token. For mobile clients the client has to be valid for the given mag-identifier
	3006134	invalid_request	CORS requirements were not met	<ul style="list-style-type: none"> - The request did not provide CORS required request headers 	Verify that the user-agent (browser) supports CORS and adds required headers to the request
	3006201	invalid_request	invalid client	<ul style="list-style-type: none"> - The given client_id or client_secret is not valid - The client_id and/or client_secret is missing - The credentials were not provided as base64 encoded value - The client is disabled - The client_id has expired (if it had a limited lifetime) 	Repeat the request using valid credentials. If the error still occurs contact the system administrator
	3006203	invalid_request	missing SSL	<ul style="list-style-type: none"> - The client did not use 'https' but 'http' 	Repeat the request using 'https'
3007/ resource_owner_session_status	3007000	invalid_request	invalid		
	3007103	invalid_request	invalid parameters	<ul style="list-style-type: none"> - The request did not include headers and/or parameters as specified for the API - The request included duplicate parameters 	Repeat the request including all required parameters and/or headers

	3007134	invalid_request	CORS requirements were not met	- The request did not provide CORS required request headers	Verify that the user-agent (browser) supports CORS and adds required headers to the request
	3007203	invalid_request	missing SSL	- The client did not use 'https' but 'http'	Repeat the request using 'https'
	3007990	invalid_request	invalid access_token	- The token has expired - The token does not exist	Request a new token
	3007991	invalid_request	invalid access_token	- The token has not been granted for the required SCOPE	Request a new token that is scoped accordingly
	3007992	invalid_request	invalid access_token	- No access_token was included in the request - An access_token was included more than once	Use the access_token either within the authorization header, as query parameter or as post body parameter
	3007993	invalid_request	invalid token	- The token is disabled which means that associated client is disabled	The token (client) has to be enabled or revoked and a new one requested.
4000/ get_client_id_filter, get_client_id_ident, get_client_id_org, get_client_id_name, get_client_id, get_all_client_id, get_client_by_ident, get_client_by_clientkey, get_client_registered_by, get_client_org, get_client_by_name_org, get_all_client	4000000	invalid_request	invalid		
	4000103	invalid_request	invalid parameters	- The request did not include headers and/or parameters as specified for the API - The request included duplicate parameters - The request included required headers or parameters but without value, empty	Repeat the request including all required parameters and/or headers

	4000204	invalid_request	invalid request	- The requester did not use mutual SSL	Repeat the request using mutual SSL
	4000205	invalid_client	invalid client certificate	- The given client certificate is unknown	Repeat the request using valid credentials. If the error still occurs contact the system administrator
	4000303	invalid_request	invalid operation	- The API was used with an invalid ending path value	Repeat the request using a valid path
4001/persist_client, persist_client_id, persist_client_and_client_id	4001000	invalid_request	invalid		
	4001103	invalid_request	invalid parameters	- The request did not include headers and/or parameters as specified for the API - The request included duplicate parameters - The request included required headers or parameters but without value, empty	Repeat the request including all required parameters and/or headers
	4001201	invalid_request	invalid client	- The given client_id references an unknown client - The given client is unknown	Repeat the request using valid credentials. If the error still occurs contact the system administrator
	4001300	invalid_request	client could not be persisted	- The 'client_id' already exist - The 'client_key' already exist - The combination of 'name' and 'organization' already exists	Repeat the request using unique values.
	4001301	invalid_request	client could not be persisted	- The given value for 'persist_type' is invalid	Repeat the request using a valid value for 'persist_type'
	4001303	invalid_request	invalid operation	- The API was used with an invalid ending path value	Repeat the request using a valid path
4002/delete_client, revoke_client_id	4002000	invalid_request	invalid		

	4002103	invalid_request	invalid parameters	<ul style="list-style-type: none"> - The request did not include headers and/or parameters as specified for the API - The request included duplicate parameters - For 'revoke': The request included 'client_ident' and 'client_key' - The request included required headers or parameters but without value, empty 	Repeat the request including all required parameters and/or headers
	4002303	invalid_request	invalid operation	<ul style="list-style-type: none"> - The API was used with an invalid ending path value 	Repeat the request using a valid path
4003/ update_client, update_client_id, update_client_id_registered_by	4003000	invalid_request	invalid		
	4003103	invalid_request	invalid parameters	<ul style="list-style-type: none"> - The request did not include headers and/or parameters as specified for the API - The request included duplicate parameters - The request included required headers or parameters but without value, empty 	Repeat the request including all required parameters and/or headers
	4003303	invalid_request	invalid operation	<ul style="list-style-type: none"> - The API was used with an invalid ending path value 	Repeat the request using a valid path
4100/ persist_temporary	4100000	invalid_request	invalid		
	4100103	invalid_request	invalid parameters	<ul style="list-style-type: none"> - The request did not include headers and/or parameters as specified for the API - The request included duplicate parameters - The request included required headers or parameters that were empty 	Repeat the request including all required parameters and/or headers

	4100204	invalid_request	invalid request	- The requester did not use mutual ssl	Repeat the request using mutual SSL
	4100205	invalid_client	invalid client certificate	- The given client certificate is unknown	Repeat the request using valid credentials. If the error still occurs contact the system administrator
	4100303	invalid_request	invalid operation	- The API was used with an invalid ending path value	Repeat the request using a valid path
4101/ persist_token_oauth1	4101000	invalid_request	invalid		
	4101103	invalid_request	invalid parameters	- The request did not include headers and/or parameters as specified for the API - The request included duplicate parameters - The request included required headers or parameters that were empty	Repeat the request including all required parameters and/or headers
	4101303	invalid_request	invalid operation	- The API was used with an invalid ending path value	Repeat the request using a valid path
4102/ persist_token_oauth2	4102000	invalid_request	invalid		
	4102103	invalid_request	invalid parameters	- The request did not include headers and/or parameters as specified for the API - The request included duplicate parameters - The request included required headers or parameters that were empty	Repeat the request including all required parameters and/or headers
	4102135	invalid_request	Token limit reached	- The maximum number of access tokens has been reached for the given Client (App) and Resource Owner	Revoke any unused tokens or increase the max number of allowed access tokens
	4102303	invalid_request	invalid operation	- The API was used with an invalid ending path value	Repeat the request using a valid path

4103/ update_token_status	4103000	invalid_request	invalid		
	4103103	invalid_request	invalid parameters	- The request did not include headers and/or parameters as specified for the API - The request included duplicate parameters - The request included required headers or parameters that were empty	Repeat the request including all required parameters and/or headers
	4103303	invalid_request	invalid operation	- The API was used with an invalid ending path value	Repeat the request using a valid path
4104/ update_oauth1_token_owner	4104000	invalid_request	invalid		
	4104103	invalid_request	invalid parameters	- The request did not include headers and/or parameters as specified for the API - The request included duplicate parameters - The request included required headers or parameters that were empty	Repeat the request including all required parameters and/or headers
	4104303	invalid_request	invalid operation	- The API was used with an invalid ending path value	Repeat the request using a valid path
4105/ update_oauth1_token_verifier	4105000	invalid_request	invalid		
	4105103	invalid_request	invalid parameters	- The request did not include headers and/or parameters as specified for the API - The request included duplicate parameters - The request included required headers or parameters that were empty	Repeat the request including all required parameters and/or headers

	4105125	invalid_request	invalid verifier	- the given token already has an associated verifier	Repeat the authorization flow and set the verifier once only
	4105303	invalid_request	invalid operation	- The API was used with an invalid ending path value	Repeat the request using a valid path
4106/ revoke_oauth_token	4106000	invalid_request	invalid		
	4106103	invalid_request	invalid parameters	- The request did not include headers and/or parameters as specified for the API - The request included duplicate parameters - The request included required headers or parameters that were empty	Repeat the request including all required parameters and/or headers
	4106303	invalid_request	invalid operation	- The API was used with an invalid ending path value	Repeat the request using a valid path
4107/ delete_oauth_token	4107000	invalid_request	invalid		
	4107103	invalid_request	invalid parameters	- The request did not include headers and/or parameters as specified for the API - The request included duplicate parameters - The request included required headers or parameters that were empty	Repeat the request including all required parameters and/or headers
	4107303	invalid_request	invalid operation	- The API was used with an invalid ending path value	Repeat the request using a valid path
4108/ disable_oauth_token	4108000	invalid_request	invalid		

	4108103	invalid_request	invalid parameters	- The request did not include headers and/or parameters as specified for the API - The request included duplicate parameters - The request included required headers or parameters that were empty	Repeat the request including all required parameters and/or headers
	4108303	invalid_request	invalid operation	- The API was used with an invalid ending path value	Repeat the request using a valid path
4109/ get_oauth_token, get_oauth_token_param, get_oauth_token_cid_ro, get_oauth_token_status_ro	4109000	invalid_request	invalid		
	4109103	invalid_request	invalid parameters	- The request did not include headers and/or parameters as specified for the API - The request included duplicate parameters - The request included required headers or parameters that were empty	Repeat the request including all required parameters and/or headers
	4109303	invalid_request	invalid operation	- The API was used with an invalid ending path value	Repeat the request using a valid path
4110/ get_temporary_token_t, get_temporary_token_t_v, get_temporary_token_cid_ro	4110000	invalid_request	invalid		
	4110103	invalid_request	invalid parameters	- The request did not include headers and/or parameters as specified for the API - The request included duplicate parameters - The request included required headers or parameters that were empty	Repeat the request including all required parameters and/or headers

	4110303	invalid_request	invalid operation	- The API was used with an invalid ending path value	Repeat the request using a valid path
4111/register_jwt	4111000	invalid_request	invalid		
	4111103	invalid_request	invalid parameters	- The request did not include headers and/or parameters as specified for the API - The request included duplicate parameters - The request included required headers or parameters that were empty	Repeat the request including all required parameters and/or headers
	4111303	invalid_request	invalid operation	- The API was used with an invalid ending path value	Repeat the request using a valid path
4112/lookup_jwt	4112000	invalid_request	invalid		
	4112103	invalid_request	invalid parameters	- The request did not include headers and/or parameters as specified for the API - The request included duplicate parameters - The request included required headers or parameters that were empty	Repeat the request including all required parameters and/or headers
	4112303	invalid_request	invalid operation	- The API was used with an invalid ending path value	Repeat the request using a valid path
4113/remove_jwt	4113000	invalid_request	invalid		
	4113103	invalid_request	invalid parameters	- The request did not include headers and/or parameters as specified for the API - The request included duplicate parameters - The request included required headers or parameters that were empty	Repeat the request including all required parameters and/or headers

	4113303	invalid_request	invalid operation	- The API was used with an invalid ending path value	Repeat the request using a valid path
4200/delete_expired_session	4200000	invalid_request	invalid		
	4200103	invalid_request	invalid parameters	- The request did not include headers and/or parameters as specified for the API - The request included duplicate parameters - The request included required headers or parameters that were empty	Repeat the request including all required parameters and/or headers
	4200204	invalid_request	invalid request	- The requester did not use mutual SSL	Repeat the request using mutual SSL
	4200205	invalid_client	invalid client certificate	- The given client certificate is unknown	Repeat the request using valid credentials. If the error still occurs contact the system administrator
	4200303	invalid_request	invalid operation	- The API was used with an invalid ending path value	Repeat the request using a valid path
4201/delete_session	4201000	invalid_request	invalid		
	4201103	invalid_request	invalid parameters	- The request did not include headers and/or parameters as specified for the API - The request included duplicate parameters - The request included required headers or parameters that were empty	Repeat the request including all required parameters and/or headers
	4201303	invalid_request	invalid operation	- The API was used with an invalid ending path value	Repeat the request using a valid path
4202/get_session	4202000	invalid_request	invalid		

	4202103	invalid_request	invalid parameters	- The request did not include headers and/or parameters as specified for the API - The request included duplicate parameters - The request included required headers or parameters that were empty	Repeat the request including all required parameters and/or headers
	4202303	invalid_request	invalid operation	- The API was used with an invalid ending path value	Repeat the request using a valid path
4203/ create_session	4203000	invalid_request	invalid		
	4203103	invalid_request	invalid parameters	- The request did not include headers and/or parameters as specified for the API - The request included duplicate parameters - The request included required headers or parameters that were empty	Repeat the request including all required parameters and/or headers
	4203303	invalid_request	invalid operation	- The API was used with an invalid ending path value	Repeat the request using a valid path
5000/ validate_client	5000103	invalid_request	invalid parameters	- The request did not include the client_key parameter	Repeat the request including all required parameters and/or headers
	5000115	invalid_scope	invalid scope was requested	- the given SCOPE for this client is not valid	Repeat the request using valid scope values
	5000126	invalid_request	invalid status	- the expected client status is not the actual one	Verify that the client has the expected status
	5000127	invalid_request	expired client id	- The client id has expired - The client id has been removed due to its expiration	Check if your requested values are correct
	5000128	invalid_client	invalid client name	- The client_name is invalid	Check if your requested values are correct

	5000129	invalid_request	invalid environment	- The environment is not valid - The client has been configured for a different one	Check if the environment for the client id is configured as expected
	5000130	invalid_request	invalid client type	- The client type is not the actual one	Check if the client type for the client is configured as expected
	5000201	invalid_request	invalid client	- The client_id is a master-key - The client_secret is invalid	Check if your requested values are correct
	5000204	invalid_request	invalid request	- The requester did not use mutual SSL	Repeat the request using mutual SSL
	5000205	invalid_client	invalid client certificate	- The given client certificate is unknown	Repeat the request using valid credentials. If the error still occurs contact the system administrator
5001/ validate_token	5001000	invalid_request	invalid		
	5001103	invalid_request	invalid parameters	- The request did not include headers and/or parameters as specified for the API - The request included duplicate parameters - The request included required headers or parameters that were empty	Repeat the request including all required parameters and/or headers
	5001204	invalid_request	invalid request	- The requester did not use mutual SSL	Repeat the request using mutual SSL
	5001205	invalid_client	invalid client certificate	- The given client certificate is unknown	Repeat the request using valid credentials. If the error still occurs contact the system administrator
	5001990	invalid_request	invalid access_token	- The token has expired - The token does not exist	Request a new token
	5001991	invalid_request	invalid access_token	- The token has not been granted for the required SCOPE	Request a new token that is scoped accordingly

	5001992	invalid_request	invalid access_token	<ul style="list-style-type: none"> - No access_token was included in the request - An access_token was included more than once 	Use the access_token either within the authorization header, as query parameter or as post body parameter
	5001993	invalid_request	invalid token	<ul style="list-style-type: none"> - The token is disabled which means that associated client is disabled 	The token (client) has to be enabled or revoked and a new one requested.
5002/ validate_refresh_token	5002103	invalid_request	invalid parameters	<ul style="list-style-type: none"> - The request did not include headers and/or parameters as specified for the API - The request included duplicate parameters - The request included required headers or parameters that were empty 	Repeat the request including all required parameters and/or headers
	5002115	invalid_scope	invalid scope was requested	<ul style="list-style-type: none"> - No scope value matched a valid one for this refresh_token 	Repeat the request using valid scope values
	5002117	unauthorized_client	unauthorized client for this request	<ul style="list-style-type: none"> - The refresh_token has been issued to a different client_id 	Repeat the request using a valid client
	5002204	invalid_request	invalid request	<ul style="list-style-type: none"> - The requester did not use mutual SSL 	Repeat the request using mutual SSL
	5002205	invalid_client	invalid client certificate	<ul style="list-style-type: none"> - The given client certificate is unknown 	Repeat the request using valid credentials. If the error still occurs contact the system administrator
	5002990	invalid_request	invalid access_token	<ul style="list-style-type: none"> - The token has expired - The token does not exist 	Request a new token
	5002993	invalid_request	invalid token	<ul style="list-style-type: none"> - The token is disabled which means that associated client is disabled 	The token (client) has to be enabled or revoked and a new one requested.

5003/ token_revocation	5003103	invalid_request	invalid parameters	<ul style="list-style-type: none"> - The request did not include headers and/or parameters as specified for the API - The request included duplicate parameters - The request included required headers or parameters that were empty 	Repeat the request including all required parameters and/or headers
	5003117	unauthorized_client	unauthorized client for this request	<ul style="list-style-type: none"> - The token has been issued to a different client_id 	Repeat the request using a valid client
	5003118	unsupported_token_type	unsupported token type	<ul style="list-style-type: none"> - The values for token_type_hint must either be 'access_token' or 'refresh_token' 	Repeat the request using the correct value for token_type_hint
	5003204	invalid_request	invalid request	<ul style="list-style-type: none"> - The requester did not use mutual SSL 	Repeat the request using mutual SSL
	5003205	invalid_client	invalid client certificate	<ul style="list-style-type: none"> - The given client certificate is unknown 	Repeat the request using valid credentials. If the error still occurs contact the system administrator
5004/ validate_id_token, create_id_token	5004103	invalid_request	invalid parameters	<ul style="list-style-type: none"> - The request did not include headers and/or parameters as specified for the API - The request included duplicate parameters - The request included required headers or parameters that were empty 	Repeat the request including all required parameters and/or headers
	5004117	unauthorized_client	unauthorized client for this request	<ul style="list-style-type: none"> - The token has been issued to a different recipient 	Repeat the request using a valid client or request a valid JWT
	5004120	invalid_request	invalid id_token	<ul style="list-style-type: none"> - The id_token has expired - The id_token has missing claims 	Repeat the request including a valid id_token

	5004121	invalid_request	invalid JWT	<ul style="list-style-type: none"> - The JWT has an invalid signature - The JWT has been created with a different shared secret 	Repeat the request using a valid JWT
	5004122	invalid_request	invalid JWT/ id_token request	<ul style="list-style-type: none"> - If an access_token was given a nonce has to be available also - If the id_token had other missing claims - If the JWT's signature could not be created 	Repeat the request including valid values for creating the JWT/ id_token
	5004131	invalid_request	invalid id_token request	<ul style="list-style-type: none"> - If an access_token was given, a nonce must be available 	Repeat the request including valid values for creating the id_token
	5004204	invalid_request	invalid request	<ul style="list-style-type: none"> - The requester did not use mutual SSL 	Repeat the request using mutual SSL
	5004205	invalid_client	invalid client certificate	<ul style="list-style-type: none"> - The given client certificate is unknown 	Repeat the request using valid credentials. If the error still occurs contact the system administrator

Database Maintenance

The **Persistence Layer: MySQL or Oracle** solution kit installed with the OTK Solution Kit creates a maintenance folder containing pre-configured scheduled tasks. The tasks replace the need to run cron jobs to clean-up the database.

NOTE

These database maintenance tasks are not required for Cassandra databases. Cassandra databases mark expired data with tombstones and schedule the data for subsequent removal.

The tasks remove the following expired items from OTK related database tables:

- oauth_client_key
- oauth_initiate
- oauth_token
- oauth_id_token

No configuration of these recurring tasks is required. They are pre-configured to run at scheduled intervals.

Manage Scheduled Tasks					
Job Type	Job Name	Policy Name	Schedule	Node	Status
Recurring	OTK Database Maintenance ...	OTK Database Maintenance - token	0 */13 * * * ?	All	Scheduled
Recurring	OTK Database Maintenance ...	OTK Database Maintenance - sessions	0 */7 * * * ?	All	Scheduled
Recurring	OTK Database Maintenance ...	OTK Database Maintenance - id_token	0 */29 * * * ?	All	Scheduled
Recurring	OTK Database Maintenance ...	OTK Database Maintenance - Client	0 */31 * * * ?	All	Scheduled

Configure Database Maintenance Tasks

You can modify how often a task is run, or add new tasks.

To configure the tasks for an OTK database (MySQL or Oracle):

1. In the Policy Manager tool bar, navigate to **Tasks, Global Settings, Manage Scheduled Tasks**. A list of currently scheduled tasks appears.
2. Select any of the existing tasks.
3. Click **Edit** and modify the scheduled task properties. Click **Add** to create a new task.
4. Click **OK**.
5. Click **Close**.

Scheduled Task Properties

Setting	What you should know...
Job Name	This name identifies the scheduled task.
Gateway Node	Choose to run the task on All Nodes or One Node .
Policy	Choose a policy to run.
Execution Time	Choose whether the policy is recurring or executes one time only. For recurring policies, select a simple time interval. Clearing the "Enable" check box lets you temporarily suspend the recurring job, while keeping ability to run one-time jobs.

Setting	What you should know...
Day of Week	Select which day(s) of the week the schedule applies. If no days are selected, then the task can run on any day of the week. Note: Selecting any day of the week automatically disables the "Day" setting under Advanced.
Execute policy with user	To execute the policy using the credentials of a specified user, set via the Select User button.

OpenID Connect Implementation

OpenID Connect is installed by default with the CA API Gateway. The OpenID Connect endpoints are installed through the OAuth Solution Kit.

Import Certificates

To complete the installation, import SSL certificates. Importing SSL certificates permits the Gateway to be used as a client.

1. Select **Tasks, Certificates, Keys and Secrets, Manage Certificates**. The Manage Certificates dialog is displayed.
2. Select **Add**. The Add Certificate Wizard starts.
3. Select **Retrieve via SSL Connection** and type: **https://localhost:8443/**
4. Click **Next**.
5. You are warned about a hostname mismatch, Click **Accept**. The Certificate details are displayed.
6. Click **Next**.
7. Select the following check boxes:
 - Outbound SSL Connections**
 - Signing Certificates for Outbound SSL Connections**
 - Signing Client Certificates**
8. Click **Finish** to complete the wizard.
9. Restart the Gateway:


```
service ssg restart
```

Configure the Callback URL of the Test Client

To configure the callback URL through the OAuth Manager:

1. Open a browser and navigate to:


```
https://<hostname>:8443/<instanceMod>/oauth/manager
```

 The *hostname* is the hostname of the gateway. For example: `gateway.com`
 The optional *instanceMod* value distinguishes between multiple gateway instances on the same server.
2. Provide a username and password. The type of access you are granted depends on your user role.
3. Click **Clients**.
4. In `client_ident` column, locate the **OpenID Connect Basic Client Profile** client and click **List Keys**. The key details are displayed.
5. Click **Edit** and then replace the **Callback URL** field with the protocol, hostname, port, and optional instance modifier of your gateway.
 Example: `https://myGateway.com:8443/instanceMod`
6. Click **Save**.

Run the Test Client

1. Open a browser and connect to one of the following URLs to open the OpenID Connect Test Client.

<code><Gateway>:8443/oauth/v2/client/bcp</code>	Basic Client Profile
<code><Gateway>:8443/oauth/v2/client/icp</code>	Implicit Client Profile

2. Click **send**. The browser is redirected to the authorization endpoint.
3. Provide the credentials of any user listed in the Internal Identity Provider of the Gateway. Click **Login**.

4. Click **Grant** to continue.
Granting access allows the client to access to protected resources and personal information. Personal information is accessed through the /userinfo OpenID Connect endpoint.
The browser is redirected back to the client. The client receives the following tokens:
 - access_token (allows the client to access the personal information of the user)
 - refresh_token
 - id_token
5. Click **Claims** to access the "/userinfo" endpoint.
If the gateway is installed and working correctly, a JSON message containing several claims is returned.

Open ID Connect Implementation Details

The following is required for Open ID Authentication:

- The Gateway must be configured for token_type BEARER.
OpenID Connect is not available with other token types.
- The "DMZ, OAuth 2.0 and OpenID Connect endpoints" solution kit is installed.

The implementation uses a database to persist all data. The client stores the id_token using a "Store to Cache" assertion.

The following sections relate to OpenID Connect Implementation:

Implicit Flow

In the Implicit Flow, the access token and ID token are returned directly to the client, which may expose them to the user and user applications. The Authorization Server does not perform Client Authentication.

The Implicit Flow is as follows:

1. The client prepares an authentication request containing the desired request parameters.
2. The client sends the request to the Authorization Server.
3. The Authorization Server authenticates the user.
4. The Authorization Server obtains user consent/authorization.
5. The Authorization Server sends the user back to the client with an access token and, if requested, an ID token.
6. The client validates the ID token and retrieves the subject identifier of the user.

Related response types:

- id_token
- id_token token

Authorization Code Flow

The Authorization Code Flow returns an Authorization Code to the client, which can then exchange it for an ID token and an Access Token directly. No tokens are exposed. The Authorization Server can also authenticate the Client before exchanging the Authorization Code for an Access Token.

The Authorization Code Flow is as follows:

1. The client prepares an authentication request containing the desired request parameters.
2. The client sends the request to the Authorization Server.
3. The Authorization Server authenticates the user.
4. The Authorization Server obtains user consent/authorization.

5. The Authorization Server sends the user back to the client with an Authorization Code.
6. The client requests a response using the Authorization Code at the Token Endpoint.
7. The client receives a response that contains an ID Token and Access Token in the response body.
8. The client validates the ID token and retrieves the user's Subject Identifier.

Related response types:

- code

Hybrid Flow

When using the Hybrid Flow, some tokens are returned from the Authorization Endpoint and others are returned from the Token Endpoint.

The Hybrid Flow is as follows:

1. The client prepares an Authentication Request containing the desired request parameters.
2. The client sends the request to the Authorization Server.
3. The Authorization Server Authenticates the user.
4. The Authorization Server obtains user consent/authorization.
5. The Authorization Server sends the user back to the client with an Authorization Code and, depending on the Response Type, one or more additional parameters.
6. The client requests a response using the Authorization Code at the Token Endpoint.
7. The client receives a response that contains an ID Token and Access Token in the response body.
8. The client validates the ID Token and retrieves the user's Subject Identifier.

Related response types:

- code token
- code id_token
- code id_token token

Authentication Request Parameters

The following parameters can be included in the URL of an authorization request:

Request Parameters	Values	Notes
client_id		Required. The OAuth 2.0 client identifier of the requesting client. Must be known to the authorization server.
state	String value	A string ignored by the server that can be used to track the session. Returned as received.
nonce		String value used to associate a Client session with an ID Token, and to mitigate replay attacks. Appears within the id_token. For example: 124512:wot12hs5h Required if response_type = token id_token.

display	Supported values: <ul style="list-style-type: none"> page social_login 	Specifies how the Authorization Server displays the authentication and consent user interface pages. Default: page – An HTML page view is displayed to request authentication and consent. social_login – Supported in MAG installations only. Provides options to authenticate using social login credentials such as Facebook. The response is a JSON message rather than an HTML page. Forwarded to the /authorize/login API.
id_token_hint		Default: Contains the previously issued id_token representing the current resource_owner. Required when prompt=none. Forwarded to the /authorize/login API with action=login
prompt	Supported values: <ul style="list-style-type: none"> none login consent 	A space separated list of values. The prompt parameter specifies whether the Authorization Server asks the end user for re-authentication and consent. It is used to make sure the user is still present for the current session or to bring attention to the request. Default: login consent none – The server does not request user authentication or consent if the user is currently logged in and the client has previously received requested grants. Cannot be used with any other value, otherwise an error is returned. login – The user is prompted to provide login credentials (username/password) for re-authentication. If re-authentication fails, an error is returned. consent – The user is prompted to provide consent. If no consent is provided, an error is returned.
acr_values		Indicates which Authentication Context Class Reference (acr) classes are acceptable for the user authentication. Based on the requested acr claim value, the Authorization Server can set thresholds for allowing authentication, requesting re-authentication, or denying authentication. The value is forwarded to the /authorize/login API where the accepted values are defined and any thresholds can be set. Values appear in order of preference.
scope	openid	Required for all OpenID requests. The client must have been registered with "openid" as a valid scope value. A client can request any additional scope, but only registered scopes will be granted.

response_type	One of the following: <ul style="list-style-type: none"> code token id_token 	Required. Determines the authorization processing flow. code – indicates an Authorization Code is returned (authorization flow). token id_token – indicates an access token is returned (implicit flow). When response_type = token id_token, nonce is required.
redirect_uri		The response is sent to this URI.
MSSO Related		
mag-identifier		Represents a valid registered mobile device. Only supported when response_type = code (authorization flow) Required for MAG clients.
PKCE Parameters- See Configure PKCE Support .		

Related Endpoints

Endpoint	Notes
/auth/oauth/v2/authorize	Supports OpenID Connect parameters 'id_token_hint', 'acr_values' For PKCE support: Checks that code_challenge is present. If present, the code_challenge_method must exist. Adds code_challenge and code_challenge_method to the session created with each authorization request.
/auth/oauth/v2/authorize/login	Handles the login process during OAuth authentication for response types
/auth/oauth/v2/authorize/consent	Handles the consent process during OAuth authentication for response types
/auth/oauth/v2/token	Issues an access token. For PKCE support: Uses the authorization_code to look up the associated session. If code_challenge exists, code_verifier must be provided as an input. Recalculates the code_challenge, from the code_verifier value using the persisted code_challenge method. Compares the calculated code_challenge to the persisted code_challenge.
/oauth/validation/validate/v2/idtoken	Used when a client uses the "code" flow and when a user session must be validated.
/openid/connect/v1/client/*	This endpoint implements the OpenID Connect test clients. The policy can be modified to change the behavior of the test clients. Client types are "Basic Client Profile" and "Implicit Client Profile".

/openid/connect/v1/userinfo	<p>This endpoint returns a JSON object containing claims defined by OpenID Connect. The content of the result depends on the granted SCOPE. The access_token used must be granted for the SCOPE openid.</p> <p>By default, this endpoint returns the same result for any request except for the "sub" claim and the overall number of returned claims. The number of claims depends on granted/ requested SCOPE values. To change this default behavior, modify the policy to retrieve user specific values from an LDAP identity provider.</p>
-----------------------------	---

NOTE

When using the Internet Explorer browser, a request to /openid/connect/v1/userinfo may cause the browser to present a download menu. Internet Explorer does not support the application/json content type.

Generate and Validate an ID Token

This topic explains ID Token configuration, generating ID tokens with various signing algorithms, and validating of custom ID tokens.

The OpenID Connect implementation contains additional assertions that are not found in the core system. In these assertions, all text fields also support clusterwide properties and context variables. Any errors that are generated by these assertions are described in detail in the Gateway Audit Events window.

To add the assertions to a custom policy, drag-and-drop the assertion from the Message Validation/Transformation category in the **Assertion** tab.

Related tasks include:

NOTE

See the [CA API Gateway documentation](#) for additional information about the following topics:

- Encode Json Web Token Assertion
- Decode Json Web Token Assertion
- Private Key Properties

ID Token Configuration

The Encode Json Web Token assertion generates an id_token.

The **OTK id_token Generation** policy generates an id_token and exposes it as a JWT. The read-only policy is located in OTK/Policy Fragments/manage/id_token.

The **OTK id_token configuration** policy contains context variables that are used for OTK id_token generation.

To set custom values for these variables, cut and paste the Set Context Variable assertions from the **OTK id_token configuration** policy into the editable **#OTK id_token configuration** policy.

Context Variable	Default Value	Notes
iss	https://{gateway.cluster.hostname}:8443	By default, the hostname of the Gateway is used to create the iss URL value. Setting the iss value is a post-installation task for dual gateways. See Post-Installation Tasks .

id_token_signing_algorithm	HS256	The value reflects the algorithm that the OTK server uses to sign the JWT. The policy logic checks to see if the algorithm is HS256 or not. Set to one of the following values: <ul style="list-style-type: none"> • HS256 – default • any other value representing your chosen algorithm. For example, RS256
subject_type	pairwise	Set to one of the following values: <ul style="list-style-type: none"> • public – provides the same sub value to all clients. Sub value is the resource owner name. • pairwise – provides unique PPID for sub identifier for each user, client, iss combination. Used for Dynamic Client Registration. Use with the <code>sector_identifier_uri</code> is not yet supported. For client registered through OpenID Connect Dynamic Registration, the registered <code>subject_type</code> will be used instead of the context variable.

Generate an ID Token Using HS256 as a Signing Algorithm

HS256 is the default value for the `id_token_signing_algorithm` Context Variable set in the **#OTK id_token Configuration** policy. Other than setting the `iss`, no additional configuration is required.

The HS256 signing algorithm is used for confidential OAuth clients. The shared secret is the `client_secret` which is known by both the client and server. The secret is used as the key for both generating and validating the signature.

Generate an ID Token Using RS256 as a Signing Algorithm

RS256 is the default custom algorithm. Any algorithm other than HS256 is considered a custom algorithm. Custom algorithms are configured in the **OTK id_token Signing Algorithm - CUSTOM** policy.

By default, an algorithm setting of RS256 uses the default SSL key on the Gateway for signing. However, we recommend that you create a new dedicated private key to sign `id_tokens`.

The private key that generates the signature is associated with a trusted public certificate that validates the signature. You create the private key, then create the trusted certificate.

1. Open the **#OTK id_token configuration** policy.
2. Add a Set Context Variable assertion with the name: `id_token_signing_algorithm`
3. For **Expression** type: RS256. Select **OK**.
4. **Save and Activate**.
5. Create a private key to sign the `id_token`.
 - a. Go to Tasks > Certificate, Keys, and Secrets > Manage Private Keys. The list of existing private keys appears.
 - b. Select **Create**.
 - c. Provide the **Alias** (name) and select **Create**. Optionally, select the Advanced tab and select a **Signature Hash**. For more information about creating a private key, see the Private Key Properties topic in the [CA API Gateway](#) documentation.
6. Create the trusted certificate with the same name as the private key.
 - a. Go to Tasks > Certificate, Keys, and Secrets > Manage Certificates.

- b. Select **Add**.
 - c. Select **Import from Private Key's Certificate Chain**.
Select **Next**.
 - d. The certificate details appear.
Select **Next**.
 - e. Select the top three certificate usage options.
Select **Next**.
 - f. Select **Certificate is a Trust Anchor**.
Select **Finish**.
7. Open the **OTK id_token Signing Algorithm - CUSTOM** policy. Locate the policy in OTK/Customizations/id_token.
 8. For both occurrences of the Encode Json Web Token assertion perform the following tasks:
 - a. Double-click the Encode Json Web Token assertions.
 - b. Select the JWS tab.
 - c. For **Private Key**, select the private key you created for signing id_tokens. Use the same value for both assertions.
 9. In the policy, find the assertion that sets the shared_secret value. Assign the name of the certificate that is associated with the private key. The name is the same as the private key.
 10. **Save and Activate**

Generate an ID Token Using Any Other Signing Algorithm

HS256 is the default algorithm. RS256 is the default custom algorithm. Supported custom algorithms are HS384, HS512, ES256, ES384, ES512.

If any other signing algorithm is used, you must implement branches within the **OTK id_token Signing Algorithm - CUSTOM** policy to check for the algorithm. Compare the variable against the id_token_signing_algorithm Context Variable.

To use a custom signing algorithm:

1. Open the **#OTK id_token Configuration** policy.
2. Set the id_token_signing_algorithm variable to the name of your alternate algorithm. **Save and Activate**.
3. Open the **OTK id_token Signing Algorithm - CUSTOM** policy.
4. For both occurrences of the Encode Json Web Token assertion perform the following tasks:
 - a. Double-click the Encode Json Web Token assertions.
 - b. Configure JWS properties for the custom algorithm.
5. Select **OK**.
6. **Save and Activate**.

Validate an ID Token

The Decode Json Web Token assertion validates the signature of an id_token.

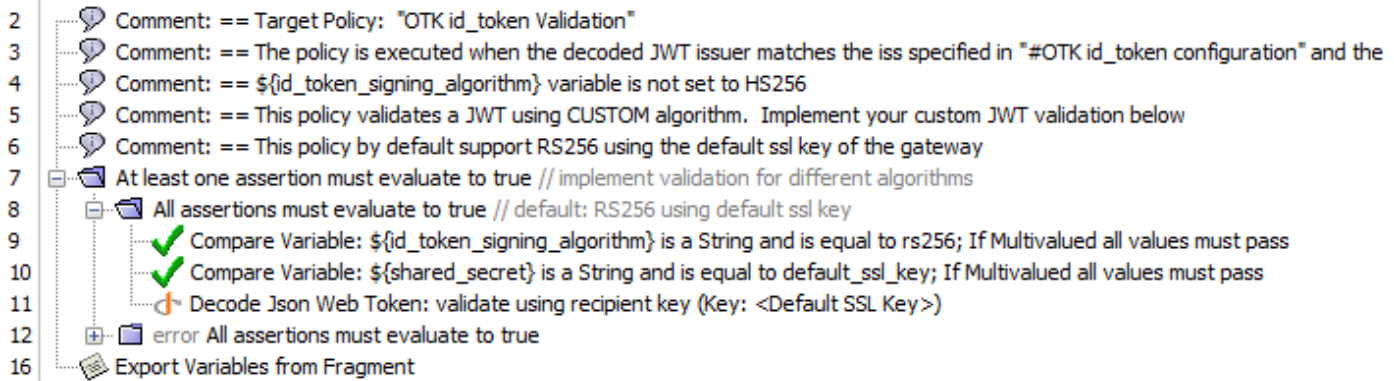
Validating Custom id_tokens (JWT) with a non-HS256 Algorithm

The OTK id_token validation - CUSTOM policy is executed when the following requirements are met:

- The decoded JWT issuer must match the iss value specified the #OTK id_token Configuration policy.
- The id_token_signing_algorithm Context Variable is set to anything other than HS256 in the #OTK id_token Configuration policy.

To validate an id_tokens with non-HS256 Algorithm:

1. Open the **OTK id_token Validation - CUSTOM** policy. The policy contains a branch that checks if the id_token_signing_algorithm is RS256 and uses the default SSL key of the Gateway as the shared



secret.

RS256, make edits in the existing branch. For other custom algorithms, create another branch for validation where the Compare Variable for `${id_token_signing_algorithm}` is equal your custom algorithm name.

2. Double-click the Compare Variable assertion for the `${shared_secret}` variable. Unless you are using the default SSL key for the Gateway as your shared secret, you must edit this field. Provide the name of the private key you created as the comparison value.
3. Double-click the Decode Json Web Token assertion and configure properties that are used for validation.

Edit the Compare Variable assertions to reflect your customization.

Suggested settings:

Validation Method – Using Recipient Key From List **Recipient Key** – Select the private key that you created

4. Select **OK** to close the assertion properties.
5. **Save and Activate.**

Validating Custom id_tokens (JWT) Issued by a Third Party

The OTK id_token Validation - CUSTOM ISS policy is executed when the decoded JWT issuer **does not** match the iss value specified in the #OTK id_token Configuration policy.

Tasks to complete:

- Create a branch in the policy specific to your custom iss.
- If the `id_token` is signed using RS256, import the certificate of the third party as a trusted certificate to validate the JWT.

Follow the comments in the **OTK id_token Validation - CUSTOM ISS** policy.

OpenID Connect Discovery

The OpenID Connect Discovery endpoint provides a client with configuration details about the OpenID Connect Authorization Server. The client makes an HTTP GET call to the discovery endpoint: **`/.well-known/openid-configuration`**. A discovery document is returned containing the OpenID Connect implementation details.

Each configuration detail is known as a claim. Claims include support capabilities and OAuth 2.0 endpoint locations that allow the client to self register and interact with the Authorization server.

The discovery endpoint follows the specification that is defined at http://openid.net/specs/openid-connect-discovery-1_0.html#ProviderMetadata.

For the swagger file describing the endpoint, see [OAuth Server API Endpoints](#).

Perform the following tasks related to OpenID Connect Discovery:

Retrieve the Discovery Document

The discovery endpoint returns discovery document as a JSON object:

```
https://<example.ca.com>:8443/<instanceModifier>/.well-known/openid-configuration
```

Where:

- `<example.ca.com>` is the hostname of the Gateway.
- `<instanceModifier>` is any optional instance modifier (prefix) added when the OTK policies were installed.

The JSON object includes claims expressed as key-value pairs that describe the OpenID Connect Server configuration.

The following example code shows the default values for claims as set in the read-only policies.

```
{
  "issuer": "http://example.ca.com",
  "authorization_endpoint": "https://example.ca.com:8443/auth/oauth/v2/authorize",
  "token_endpoint": "https://example.ca.com:8443/auth/oauth/v2/token",
  "jwks_uri": "https://example.ca.com:8443/openid/connect/jwks.json",
  "response_types_supported": ["code", "token id_token", "token", "code id_token",
  "id_token", "code token", "code token id_token"],
  "subject_types_supported": ["pairwise"],
  "id_token_signing_alg_values_supported": ["RS256", "HS256"],
  "userinfo_endpoint": "https://example.ca.com:8443/openid/connect/v1/userinfo",
  "registration_endpoint": "https://example.ca.com:8443/openid/connect/register",
  "scopes_supported": ["openid", "email", "profile", "oidc_client_registration"],
  "claims_supported": ["sub", "iss", "auth_time", "acr", "aud", "azp", "exp", "c_hash",
  "at_hash", "nonce"],
  "grant_types_supported": ["authorization_code", "implicit", "refresh_token"],
  "acr_values_supported": ["0"],
```

```

"token_endpoint_auth_methods_supported": ["client_secret_basic", "client_secret_post",
"client_secret_jwt", "private_key_jwt"],
"display_values_supported": ["page"],
"claim_types_supported": ["normal"],
"service_documentation": "https://example.ca.com:8443/apidocs/auth/oauth/v2/swagger",
"ui_locales_supported": ["en-US"],
"response_modes_supported":["query", "fragment", "form_post"],
"userinfo_signing_alg_values_supported":["RS256", "HS256"],
}

```

Configure claims by providing custom values for the corresponding Context Variables in the #policies.

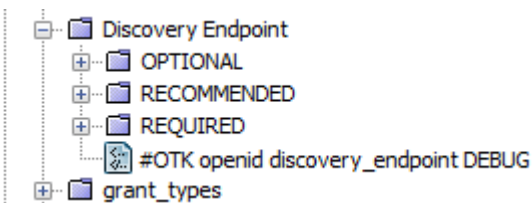
Parameter	Notes
response_types_supported	Defines the set of allowed response types for authorization requests. If a client registers dynamically through the OIDC Dynamic Registration, only registered response_types can be used, and only if they are a subset of the set of allowed response_types. See Dynamic Registration .

Configure OpenID Connect Discovery

Default settings for the OpenID Connect Discovery claims are set in read-only policies located in:

OTK/Policy Fragments/configuration/Discovery Endpoint.

Claims are defined as required, recommended, or optional by the OpenID Connect specification. These categories are reflected in the Policy Manager folder structure under OTK/Customizations/Discovery Endpoint.



Configure a Custom Value for a Claim

To configure custom values for a claim:

1. In Policy Manager, go to OTK/Customizations/Discovery Endpoint.
2. Open the OPTIONAL, RECOMMENDED, and REQUIRED sub-folders.
3. Find the #policy with the same name as the claim.
For example, to configure grant_types, open the #OTK openid grant_types_supported policy.
4. Read the comments that describe a value example, and list any restrictions.
5. Enable any disabled assertions in the #policy. By default, the Set Context Variable assertions for customizing claims are disabled.
6. Double-click the Set Context Variable assertion for the context variable.
7. Provide your custom value or values by customizing the Expression field of the context variable. Add new values or remove existing values.
8. Click **OK**.
9. **Save and Activate** the policy.

NOTE

By default, the JSON document response is cached for 300 seconds. Any changes made to #policies in the Customizations folder only take effect after the cache time expires.

For information on how to modify the cache lifetime, see [Customize Caches](#).

The cacheID is oidcDiscoveryCache.
The cacheKey is oidcDiscoveryCache.
The maxEntryAge is 300.

Disable a Claim

By default, all supported claims are exposed through the discovery document. If you do not want a claim exposed to a client, disable the claim in the #policy.

NOTE

Do not disable required claims. Required claims are located in #policies within the REQUIRED folder.

To disable a claim:

1. Open the #policy that corresponds to the name of the claim.
2. Double-click the Set Context Variable assertion for discovery.endpoint.enabled.
3. Set the Expression to **false**.
4. Click **OK**.
5. **Save and Activate** the policy.

Dynamically Register a Client

/openid/connect/register

See [Dynamic Registration](#).

Enable Logging for the Discovery Endpoint

By default, logging is turned off (false).

To enable logging, enable assertions and configure Context Variables in the #policy.

#Policy	Context Variables	Values	Notes
#OTK openid discovery_endpoint DEBUG	<pre> \${debugOnSuccess} \${debugOnError} </pre>	<pre> true true </pre>	<p>Enable detailed logs for each invocation of the OpenID Connect Discovery Endpoint. Enable detailed logs for each invocation of the OpenID Connect Discovery Endpoint that results in an error. If either variable is enabled, data is exported to capture the Discovery claims and whether they are exposed.</p>

Dynamic Registration

`/openid/connect/register`

The `/openid/connect/register` API implements the Dynamic Registration feature as specified at http://openid.net/specs/openid-connect-registration-1_0.html. Clients accessing this API can register themselves as OAuth clients for this OpenID Connect Provider.

The API generates an `access_token` with the **openid_client_registration** scope. This access token can be used at the API specified as `registration_client_uri` in the response of this API.

Supported Claims

The tables in this section list supported claims.

The following restrictions apply to dynamic registrations:

- The API accepts only those requested claim values that are specified in the JSON response of the discovery endpoint. See [OpenID Connect Discovery](#). Unsupported values are set to default values.
- Clients can only be registered once with the same values
- The following characters are not accepted for registrations and result in a failed request: `<` `>` `&` `and` `\`

OpenID Connect Specified Claims

For claim details, see the OpenID Connect specification: http://openid.net/specs/openid-connect-registration-1_0.html.

Supported Claims	Notes
<code>redirect_uris</code>	<p>The <code>redirect_uris</code> claim is required.</p> <p>The following restrictions apply to the <code>redirect_uri</code> value:</p> <ul style="list-style-type: none"> - For application_type: web, only the <code>https://</code> scheme is allowed. The hostname cannot be <code>localhost</code>. - For application_type: native, only custom schemes are allowed. The <code>http://</code> scheme is accepted only if the hostname is <code>localhost</code>. <p>- No anchors (<code>#</code>) are allowed.</p>
<code>response_types</code>	
<code>grant_types</code>	
<code>application_type</code>	
<code>contacts</code>	
<code>client_name</code>	
<code>sector_identifier_uri</code>	The expected implementation of a unique sub value for each sector identifier when pairwise subject identifiers are used is not yet supported.
<code>subject_type</code>	<p>Options include:</p> <ul style="list-style-type: none"> • <code>pairwise</code> – however, use with the <code>sector_identifier_uri</code> is not yet supported. • <code>publicl</code>

id_token_signed_response_alg	<p>The JWS algorithm that is used to sign the ID Token that is issued to this Client. The value can be either HS256 or RS256. If the value is omitted, RS256 is set.</p> <p>The algorithm value for id_token_signed_reponse_alg must match the algorithm value that is configured in ID Token Configuration section.</p>
userinfo_signed_response_alg	
token_endpoint_auth_methods	<p>Supported options include:</p> <ul style="list-style-type: none"> • client_secret_post • client_secret_basic • client_secret_jwt – To validate the JWT, the OTK has the client_secret value. • private_key_jwt – The OTK has the matching public key.
token_endpoint_auth_signing_alg	<p>The JWS algorithm that is used to sign the JWT that is used to authenticate the Client.</p> <p>One of the following values:</p> <ul style="list-style-type: none"> • HS256 • RS256 <p>Valid when the token_endpoint_auth_method is either private_key_jwt or client_secret_jwt.</p>
jwtks_uri	<p>URL for the Client JSON Web Key Set document.</p> <p>Available when token_endpoint_auth_methods is set to private_key_jwt. Cannot be passed in with</p> <pre>jwtks</pre>
jwtks	<p>Client JSON Web Key Set document, passed by value. Cannot be passed in with</p> <pre>jwtks_uri</pre>

Proprietary Claims

The following table lists supported proprietary claims.

Proprietary Claim	Notes	Default Value
scope	A custom scope. If a scope is provided, it must be supported on the server.	openid email profile openid_client_registration
organization	The organization of the requestor	The given organization of the client. If no organization is provided, the first redirect_uri is used as the value.
description	A description for this client	"Registered via OpenID Connect Dynamic Registration"
environment	The development environment of the client. For example: iOS.	ALL

Proprietary Claim	Notes	Default Value
master	This value identifies the client as being used as a "master-key". The feature is applicable to CA Mobile API Gateway clients only. A client with a master key can retrieve client credentials at /connect/client/initialize. The client_id of this client cannot be used to request an OAuth token. The client_secret generated for this client intentionally matches the value of the client_id.	false

Unsupported Claims

The following table lists claims that are not supported in the current implementation of dynamic client registration. These claims are defined as optional in the OpenID Connect specification.

Unsupported Claims	
logo_uri	request_object_signing_alg
client_uri	request_object_encryption_alg
policy_uri	request_object_encryption_enc
tos_uri	token_endpoint_auth_signing_alg
id_token_encrypted_response_alg	default_max_age
id_token_encrypted_response_enc	require_auth_time
userinfo_encrypted_response_alg	default_acr_values
userinfo_encrypted_response_enc	initiate_login_uri
	request_uris

Registration Success Response

A successful registration POST request returns a response that includes generated values such as the client_id and client_secret.

Response Example

*** Response ***

Status Line: HTTP/1.1 201 Created

Response Header: Server: Apache-Coyote/1.1

Response Header: Content-Type: application/json;charset=UTF-8

Response Header: Content-Length: 1131

Response Header: Date: Wed, 02 Aug 2017 17:49:30 GMT

Response Body:

```
{
  "client_id": "2220d66d-ccd6-4616-96c6-10c8b6cf3844",
  "client_secret": "9f03bc93-12d7-4c48-b4ab-fa17fd02931d",
  "client_secret_expires_at": 0,
  "client_id_issued_at": 1501871110,
  "registration_access_token": "b39ba851-0375-4fee-851b-2f71072e966d",
  "registration_client_uri": "https://myGateway.com:8443/openid/connect/register/2220d66d-ccd6-4616-96c6-10c8b6cf3844",
  "token_endpoint_auth_method": "client_secret_basic",
  "application_type": "web",
```

```

"redirect_uris": ["https://dynamic-client-001-test.com"],
"client_name": "https://dynamic-client-001-test.com-2017-08-04T18:25:10.852Z",
"subject_type": "pairwise",
"sector_identifier_uri": "",
"contacts": ["admin"],
"response_types": ["code"],
"grant_types": ["authorization_code"],
"id_token_signed_response_alg": "RS256",
"userinfo_signed_response_alg": "RS256",
"environment": "ALL",
"organization": "https://dynamic-client-001-test.com",
"master": false,
"description": "Registered via OpenID Connect Dynamic Registration",
"scope": "openid email profile openid_client_registration"
}

```

Retrieve Client Configuration

Using a GET request, the client configuration is available at the `/openid/connect/register/{client_id}` API.

The client uses the issued `access_token` as credentials and the issued `client_id` as the path element.

Request Example

```

Request Method: GET
Request URI: /openid/connect/register/d71d768f-a121-445e-85a4-1234abcde123
Request Header: authorization: Bearer 12328230-afd7-4303-9b06-b744462dsf06a
Request Header: User-Agent: Jakarta Commons-HttpClient/3.1
Request Header: Host: mygateway.com
Request Query: null

```

Response Example

```

{
  "subject_type": "pairwise",
  "grant_types": ["authorization_code"],
  "application_type": "web",
  "description": "Registered via OpenID Connect Dynamic Registration",
  "registration_client_uri": "https://myGateway.com:8443/openid/connect/register/2220d66d-ccd6-4616-96c6-10c8b6cf3844",
  "redirect_uris": ["https://dynamic-client-001-test.com"],
  "sector_identifier_uri": "",
  "client_id": "2220d66d-ccd6-4616-96c6-10c8b6cf3844",
  "token_endpoint_auth_method": "client_secret_basic",
  "userinfo_signed_response_alg": "RS256",
  "master": false,
  "environment": "ALL",
  "client_secret_expires_at": 0,
  "organization": "https://dynamic-client-001-test.com",
  "scope": "openid email profile openid_client_registration",
  "client_secret": "9f03bc93-12d7-4c48-b4ab-fa17fd02931d",
  "client_id_issued_at": 1501871110,
  "client_name": "https://dynamic-client-001-test.com-2017-08-04T18:25:10.852Z",
  "contacts": ["admin"],
  "response_types": ["code"],

```

```
"id_token_signed_response_alg": "RS256"
}
```

Retrieve the JSON Web Key Set (JWKS)

/openid/connect/jwks.json

The /openid/connect/jwks.json API implements the JWKS_URI as specified at http://openid.net/specs/openid-connect-discovery-1_0.html#ProviderMetadata. The endpoint returns a JWKS containing public keys that enable clients to validate a JSON Web Token (JWT) issued by this OpenID Connect Provider. The details of the response can be modified and adjusted to fit the environment.

```
{
  "keys" : [ {
    "kty" : "RSA",
    "kid" : "11234oi3mmaets3basti3nss3",
    "use" : "sig",
    "n" : "k9-F-fE4RWeyvErnyQhdbGO-468-
UYq9uoEmxZFWxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxPEbfYB0A01FAxnD5efA-6rZ
    "e" : "AQAB"
  } ]
}
```

The Authorization server can change keys. If the client fails to recognize the key ID (kid) of the JWT, the client can retrieve the new public key set from the JWKS_URI endpoint.

To configure this endpoint, see [Use a Dedicated Private Key for Signing JWT](#).

Use a Dedicated Private Key for Signing JWT

The OTK supports signing as described in the OpenID Connect specification defined in http://openid.net/specs/openid-connect-core-1_0.html#Signing.

The default SSL key is used to sign the id_token/JWT. It is recommended to use your own dedicated private key.

The following tasks describe how to sign the id_tokens with a dedicated private key that does not use the HS256 signing algorithm:

Create a Private Key

To create a private key:

1. In the Policy Manager, go to Tasks > Certificates, Keys and Secrets > Manage Private Keys.
2. Click **Create**.
3. Click **Alias** and type a name for the private key. To aid key management, consider using the same value for both alias and key ID. Key ID is assigned in a later step.
4. Click **Key type** and select an algorithm.
5. Optionally set other Basic or Advanced parameters. The key does not have to be CA capable.
6. Click **Create**.

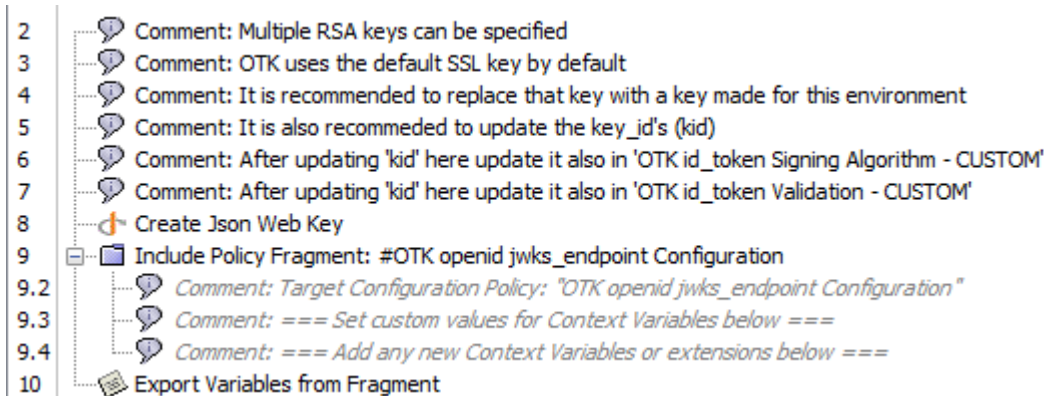
The key appears in the Manage Private Keys list.

Create a JWK From the Private Key

Now you use the private key to create the JWK used for signing the id_token/JWT.

To create a JWK:

1. Open the **OTK openid jwks_endpoint Configuration** policy in OTK/Customizations/JWKs Endpoint.



2. Copy the **Create Json Web Key** assertion. Use Right-click > Copy. This assertion instance uses the default_ssl_key to output the jwks variable.
3. Open the **#OTK openid jwks_endpoint Configuration** policy.
4. Paste the Create Json Web assertion.
5. Double-click the assertion to access properties.
6. Click **Add**.
7. For **Recipient Public Key**, select a private key.
8. Assign a **Key ID** value. The Key ID value is referenced as the "kid" in the policies and appears in the JWT header when the JWT is issued. Consider using the alias value as the Key ID value. Using the same value allows you to find the private key associated with the kid easily.
9. Copy the Key ID value. You need this value to customize policies.
10. For **Key Usage**, select **Signature**.
11. Click **OK**. The new private key appears listed.
12. Click **OK** to close the Create Json Web Key Properties dialog.
13. **Save and Activate**.

Configure the Key ID

By default, the kid value is "default SSL key" and is used in the kid header.

To configure the kid:

1. Open the OTK id_token KID Configuration policy. Locate the policy in the OTK/Policy Fragments/manage/id_token folder.
2. Copy the Set Context Variable kid assertion.
3. Open the #OTK id_token KID Configuration policy.
4. Paste the assertion.
5. Double-click the assertion.
6. Replace the default_ssl_key value with your kid value.

7. Click **OK**.
8. **Save and Activate**.

Optionally, using the same policy and #policy, you can customize how the kid_header is formed.

Generate the id_token Signed with the JWK

By default, the id_token is generated using the default SSL key. Change this behavior by modifying the **OTK id_token Signing Algorithm - CUSTOM** policy.

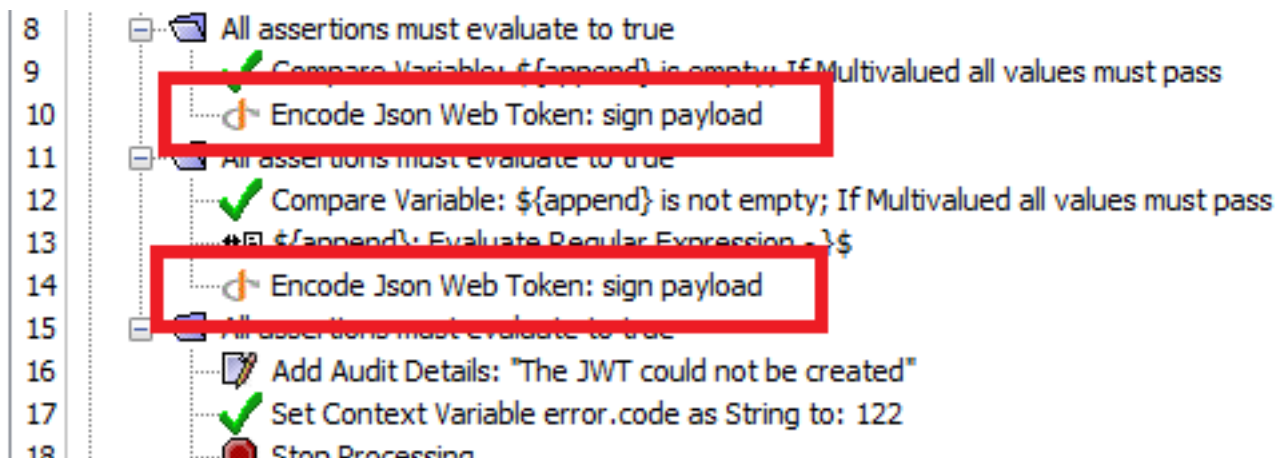
NOTE

The OTK id_token Signing Algorithm - CUSTOM policy is an extension policy in the Customizations folder that is not replaced during an upgrade.

If you have an earlier version of this policy than OTK 4.1, see [Manually Replace the Pre-OTK4.1 Version](#).

To generate the id_token from the JWK:

1. Open the **OTK id_token Signing Algorithm - CUSTOM** policy and expand the folders.
2. Perform the following steps for both **Encode Json Web Token** assertions in the policy:
 - a. Select the **JWS** tab
 - b. For **Private Key**, select the private key associated with the Key ID value. If you used the same value, this is easier.
- c. Click **OK**.



3. **Save and Activate**.

Manually Replace the Pre-OTK4.1 Version

The OTK id_token Signing Algorithm - CUSTOM policy is an extension policy in the Customizations folder that is not replaced during an upgrade.

If you have the 4.0.00 version of this policy:

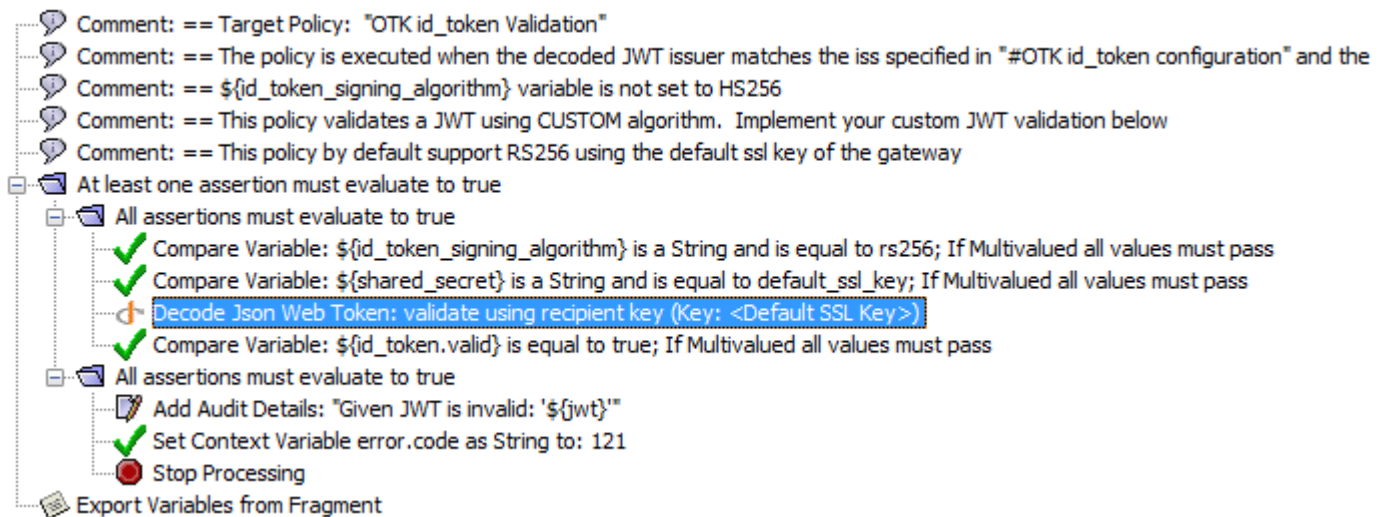
1. **Right-click** the following XML icon, select **Save link as** and download the 4.1 version of this extension policy. [#unique_147](#)
2. Delete all contents of the OTK id_token Signing Policy version 4.0.
3. Import the new 4.1 version.
4. Perform the modifications described in these instructions.

Validate the id_token with the JWK

By default, the id_token is validated using the default SSL Key.

To validate the id_token using the JWK:

1. Open the **OTK id_token Validation - CUSTOM** policy.
Expand the folders.
2. Double-click the Compare Variable for `${shared_secret}` assertion.
3. Edit the **Rule** to read: "is equal to `${kid}`"
4. Double-click the Decode Json Web Token assertion.
5. For **Recipient Key** select the alias of your private key.
6. **Save and Activate.**



OTK User Role Configuration

You can add a username to a list of known administrators.

The **OTK User Attribute Look Up Extension** policy targets the OTK User Attribute Look Up policy.

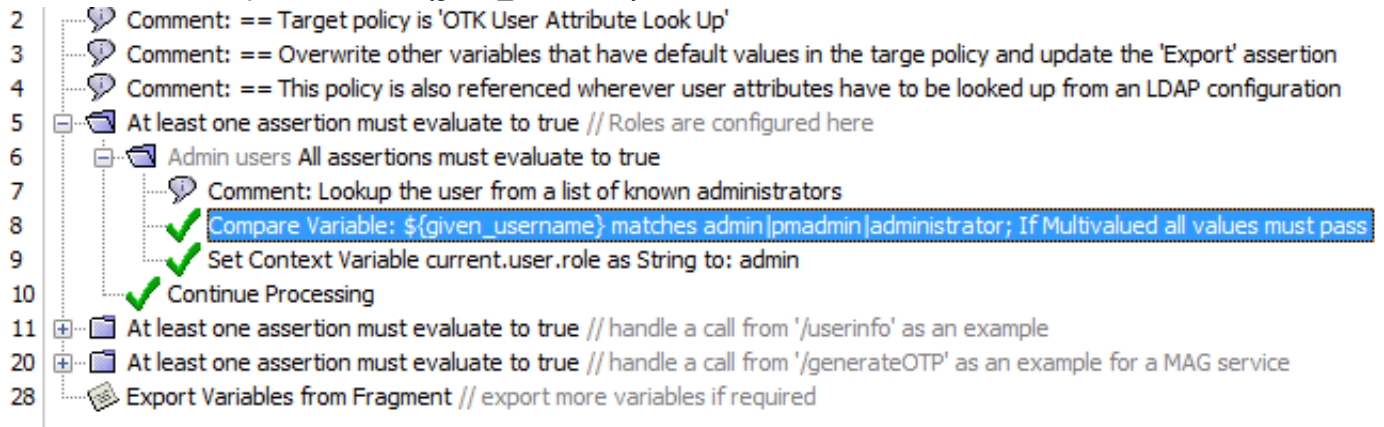
The OTK User Authentication encapsulated assertion found in OTK/Policy Fragments/authentication contains the following default user names associated with the administrator role:

- admin
- padmin
- administrator

The administrator role has a global view of clients, while the user role can see only their own clients.

To add a custom username to the administrator list:

1. Open the **OTK User Attribute Look Up Extension** policy.
2. Double-click the Compare Variable `${given_username}` assertion.



3. Select the existing rule and click **Edit**.
4. Modify the Regular Expression by adding a pipe separator character | then typing the username to be assigned the administrator role.
5. Click **OK**.
6. Click **OK** again to close the Compare Expression Properties dialog.
7. **Save and Activate**.

Tutorials

How to Install and Configure the OAuth Toolkit:

[Part 1](#)

[Part 2](#)

Documentation Legal Notice

This Documentation, which includes embedded help systems and electronically distributed materials, (hereinafter referred to as the “Documentation”) is for your informational purposes only and is subject to change or withdrawal by Broadcom at any time. This Documentation is proprietary information of Broadcom and may not be copied, transferred, reproduced, disclosed, modified or duplicated, in whole or in part, without the prior written consent of Broadcom.

If you are a licensed user of the software product(s) addressed in the Documentation, you may print or otherwise make available a reasonable number of copies of the Documentation for internal use by you and your employees in connection with that software, provided that all Broadcom copyright notices and legends are affixed to each reproduced copy.

The right to print or otherwise make available copies of the Documentation is limited to the period during which the applicable license for such software remains in full force and effect. Should the license terminate for any reason, it is your responsibility to certify in writing to Broadcom that all copies and partial copies of the Documentation have been returned to Broadcom or destroyed.

TO THE EXTENT PERMITTED BY APPLICABLE LAW, CA PROVIDES THIS DOCUMENTATION “AS IS” WITHOUT WARRANTY OF ANY KIND, INCLUDING WITHOUT LIMITATION, ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NONINFRINGEMENT. IN NO EVENT WILL CA BE LIABLE TO YOU OR ANY THIRD PARTY FOR ANY LOSS OR DAMAGE, DIRECT OR INDIRECT, FROM THE USE OF THIS DOCUMENTATION, INCLUDING WITHOUT LIMITATION, LOST PROFITS, LOST INVESTMENT, BUSINESS INTERRUPTION, GOODWILL, OR LOST DATA, EVEN IF CA IS EXPRESSLY ADVISED IN ADVANCE OF THE POSSIBILITY OF SUCH LOSS OR DAMAGE.

The use of any software product referenced in the Documentation is governed by the applicable license agreement and such license agreement is not modified in any way by the terms of this notice.

The manufacturer of this Documentation is Broadcom Inc.

Provided with “Restricted Rights.” Use, duplication or disclosure by the United States Government is subject to the restrictions set forth in FAR Sections 12.212, 52.227-14, and 52.227-19(c)(1) - (2) and DFARS Section 252.227-7014(b) (3), as applicable, or their successors.

Copyright © Broadcom. All Rights Reserved. The term “Broadcom” refers to Broadcom Inc. and/or its subsidiaries. All trademarks, trade names, service marks, and logos referenced herein belong to their respective companies.

