



# **TIBCO Cloud™ API Management - Local Edition**

## **Security Guide**

*Version 5.6.1  
February 2023*



# Contents

---

<b>Contents</b>	<b>2</b>
<b>Introduction</b>	<b>4</b>
<b>Securing the TIBCO Cloud™ API Management - Local Edition Cluster</b>	<b>5</b>
Patching Images during Image build	5
Securing Deployment	8
Authentication and Authorization	9
Developer Portal	9
Reporting	12
Port Configurations	13
Securing Cassandra	13
Securing MySQL	14
Encrypting MySQL Replication Group	16
Securing Traffic	17
HTTPS Configuration Overview	18
HTTPS Server Configuration	18
HTTPS Client Configuration	20
HTTPS Mutual Authentication	24
Securing Using OAuth	34
Securing Config UI and Dev Portal in Untethered Setup	75
Configuring LDAP and SAML with Local Edition	76
Configuring LDAP with Local Edition	76
Login with LDAP User	78
Configuring SAML with Local Edition	79
Logging in the Configuration Manager with LDAP and SAML	81
<b>TIBCO Documentation and Support Services</b>	<b>83</b>

<b>Legal and Third-Party Notices .....</b>	<b>85</b>
--	-----------

# Introduction

---

This document describes guidelines to ensure security within the various components of TIBCO Cloud™ API Management - Local Edition cluster.

# Securing the TIBCO Cloud™ API Management - Local Edition Cluster

---

The following sections describes various ways to secure the TIBCO Cloud™ API Management - Local Edition clusters.

## Patching Images during Image build

The image build process pulls libraries from various repositories and are installed through the yum utility.

To patch images during the build, choose the update libraries option.

## Creating Local Edition Docker Images

The Jenkins job, build\_docker is used to create the Local Edition Docker images.

Jenkins > Build > build\_docker

[Back to Dashboard](#)  
[Status](#)  
[Changes](#)  
[Workspace](#)  
[Build with Parameters](#)  
[Delete Project](#)  
[Configure](#)  
[Rebuild Last](#)  
[Rename](#)

## Project build\_docker

This build requires parameters:

RELEASE\_VERSION

RELEASE\_SUFFIX

TM\_CONNECTORS ☒ oauth2-jwt-authenticator  
 Selected connectors will be built into Traffic Manager container (tml-tm).

☒ BUILD\_DOCKER\_NOSQL  
 Build Cassandra docker image.

☒ BUILD\_DOCKER\_LOG  
 Build log service docker image.

☒ BUILD\_DOCKER\_SQL  
 Build MySQL docker image.

☒ BUILD\_DOCKER\_CACHE  
 Build cache docker image.

☒ BUILD\_DOCKER\_TM  
 Build Traffic Manager docker image.

☒ BUILD\_DOCKER\_CM  
 Build Cluster Manager docker image.

UPDATE\_PACKAGES

Update all packages or specified packages when docker images are built.

update\_package\_list  No file chosen  
 Click "Browse..." button to upload the CSV file which contains the list of packages to be updated.  
 This list is additional to the built-in list of packages to be updated.

☐ VERIFY\_DOCKER\_IMAGE

**Build History** [trend](#)

find

#71 Nov 27, 2019 6:58 PM

[Atom feed for all](#) [Atom feed for failures](#)

## RELEASE\_VERSION

The release version, together with build number, are used to compose the docker image tag.

In the screenshot, RELEASE\_VERSION is "5.3.0"; on the left pane of screenshot, it shows the build "#71" has been done, so Docker images in build #71 have tag "v5.3.0.GA.71".

## RELEASE\_SUFFIX

The suffix used in conjunction with the release version. For example, GA, HF1 (for Hotfix 1), and so on.

## TM\_CONNECTORS

In the **TM\_CONNECTORS** section, you can choose to build "OAuth2 JWT Authenticator" into the Traffic Manager container (tml-tm).

You can also choose the following Docker images to build:

- BUILD\_DOCKER\_NOSQL - NoSQL container (tml-nosql)
- BUILD\_DOCKER\_LOG - Log container (tml-log)
- BUILD\_DOCKER\_SQL - SQL container (tml-sql)
- BUILD\_DOCKER\_CACHE - Cache container (tml-cache)
- BUILD\_DOCKER\_TM - Traffic Manager container (tml-tm)
- BUILD\_DOCKER\_CM - Cluster Manager container (tml-cm)

**Note:** You should verify the sizes of the images that are created. They should be approximately the following sizes after creation:

- tml-nosql - 918MB
- tml-log - 1.08GB
- tml-sql - 1.65GB
- tml-cache - 848MB
- tml-tm - 857MB
- tml-cm - 1.13GB

## UPDATE\_PACKAGES

The administrator can choose:

- **Update\_Specified\_Packages:** update specified packages during build;
- **Update\_All\_Packages:** all packages will be updated during build.

### update\_package\_list:

Uploads a csv file that contains the list of packages to be updated during build. This list is applied to all Docker images.

Here is an example content of a csv file:

```
python.x86_64  
libssh2
```

```
systemd
bind-license
openssl-libs
device-mapper.x86_64
device-mapper-libs.x86_64
```

## Configuring AWS Access in the Installer

To configure access to AWS in the installer, run the `configure_aws` Jenkins job:

Complete the Project `configure_aws` dialog as follows:

- **aws\_access\_key\_id** - The AWS access key ID is required.
- **aws\_secret\_access\_key** - The AWS secret access key is required.
- **aws\_default\_region** - This is the region in which the AWS Kubernetes cluster and the Local Edition cluster are being created.
- **aws\_role\_arn** - The role Amazon Resource Name (ARN). This is required if AWS role-based access control is used. An example role ARN:

```
arn:aws:iam::123456789012:role/TIBCO/Administrator
```

## Securing Deployment

All the TIBCO Cloud™ API Management - Local Edition containers run as restricted users. It is a good practice to not run them as root or users with higher permissions.



# Authentication and Authorization

All the default passwords must be changed during and after deployment.

## Developer Portal

All the default passwords of the developer portal must be changed.

## Changing the Administrator Password

The default administrator password can be changed from the developer portal.

### Before you begin

Make sure all components are active and the cluster is up and running

```
cm ls components
Using cluster [aniket-gcp-cluster-single-host-172]
Using Zone [us-central1-a]
```

Component ID	Type	Name	Status	Last
Heartbeat Received	Host	Service Port(s)		
33319687-0658-44b9-baa4-409825c930b5	cache	cache-set-0-	ACTIVE	
0.cache-svc-0.default.svc.cluster.local				
May 24 2021 21:12:18 +0000 10.120.0.14				
11212,11211,11213,11214,11215,11216				
d5b21d3e-7d9f-4d80-80b4-2afd57177a9a	configmanager	cm-deploy-0-	ACTIVE	
5ccc99468f-m6s95.cm-svc-0.default.svc.cluster.local				
May 24 2021 21:12:09 +0000 10.120.0.11	7080			
3f9262a3-470e-4557-806e-4a23659d5fb6	logservice	log-set-0-0.log-	ACTIVE	May 24
svc-0.default.svc.cluster.local				
2021 21:12:19 +0000 10.120.0.12	24224			
da4e3f02-3d0c-45ef-9595-25ac200f3992	nosql	cass-set-0-0.cass-	ACTIVE	May 24
svc-0.default.svc.cluster.local				
2021 21:12:22 +0000 10.120.0.10	9042			
afc6a10d-1ca6-4bd8-bd51-7dd413561efe	sql	mysql-set-0-	ACTIVE	
0.mysql-svc-0.default.svc.cluster.local				
May 24 2021 21:12:12 +0000 10.120.0.13	3306			
0ab8c275-13e0-40d0-b486-96f7f64706a2	trafficmanager	tm-deploy-0-		

bc677b4c-c728g.tm-svc-0.default.svc.cluster.local	ACTIVE
May 24 2021 21:12:15 +0000 10.120.0.15 8080	

To change the default administrator password,

## Procedure

1. Launch the developer portal.

```
https://<cm_svc_ip>:8443
```

2. Navigate to **Localdev Admin > Profile** and click **Change Password**. Input the current and the new password
3. Navigate to the properties folder from the installation directory.

```
<root_dir>/docker-deploy/properties
```

4. Add new field by replacing password with new\_password as its value in the tml\_cm\_properties.json file

```
"maxIdleTime" : 300,
"api_debug_key" : "24NumbersAndOrCharacters",
"api_debug_secret" : "10NumChars",
"password" : "<new_password>"
```

5. Recompose the manifest file to generate the updated artifacts.

```
./compose.sh <manifest_json_file>
```

6. Navigate to the manifest folder and undeploy the cm pod.

```
./undeploy-cm-pod.sh
secret "cm-property" deleted
secret "cm-jks" deleted
secret "cm-crt" deleted
secret "cm-key" deleted
secret "cm-resource" deleted
deployment.apps "cm-deploy-0" deleted
```

7. Redeploy the cm pod. The new password will come in effect now.

```
./deploy-cm-pod.sh
secret/cm-property created
secret/cm-jks created
secret/cm-crt created
secret/cm-key created
secret/cm-resource created
deployment.apps/cm-deploy-0 created

cm-deploy-0-5ccc99468f-gfz7r    1/1      Running    0          10s
```

## What to do next

- Ensure all components are in active state

```
cm ls components
Using cluster [aniket-gcp-cluster-single-host-172]
Using Zone [us-central1-a]
```

Component ID	Type	Name	Status
Last Heartbeat Received	Host	Service Port(s)	
33319687-0658-44b9-baa4-409825c930b5 0.cache-svc-0.default.svc.cluster.local May 24 2021 21:12:18 +0000 10.120.0.14 11212,11211,11213,11214,11215,11216	cache	cache-set-0-	ACTIVE
d5b21d3e-7d9f-4d80-80b4-2afd57177a9a 5ccc99468f-m6s95.cm-svc-0.default.svc.cluster.local May 24 2021 21:12:09 +0000 10.120.0.11 7080	configmanager	cm-deploy-0-	ACTIVE
3f9262a3-470e-4557-806e-4a23659d5fb6 0.log-svc-0.default.svc.cluster.local May 24 2021 21:12:19 +0000 10.120.0.12 24224	logservice	log-set-0-	ACTIVE
da4e3f02-3d0c-45ef-9595-25ac200f3992 0.cass-svc-0.default.svc.cluster.local May 24 2021 21:12:22 +0000 10.120.0.10 9042	nosql	cass-set-0-	ACTIVE
afc6a10d-1ca6-4bd8-bd51-7dd413561efe 0.mysql-svc-0.default.svc.cluster.local May 24 2021 21:12:12 +0000 10.120.0.13 3306	sql	mysql-set-0-	ACTIVE

```
0ab8c275-13e0-40d0-b486-96f7f64706a2 trafficmanager tm-deploy-0-
bc677b4c-c728g.tm-svc-0.default.svc.cluster.local ACTIVE
May 24 2021 21:12:15 +0000 10.120.0.15 8080
```

- To verify the new password, login the control center using the new password.

```
https://<cm_svc_ip>:8443/admin
```

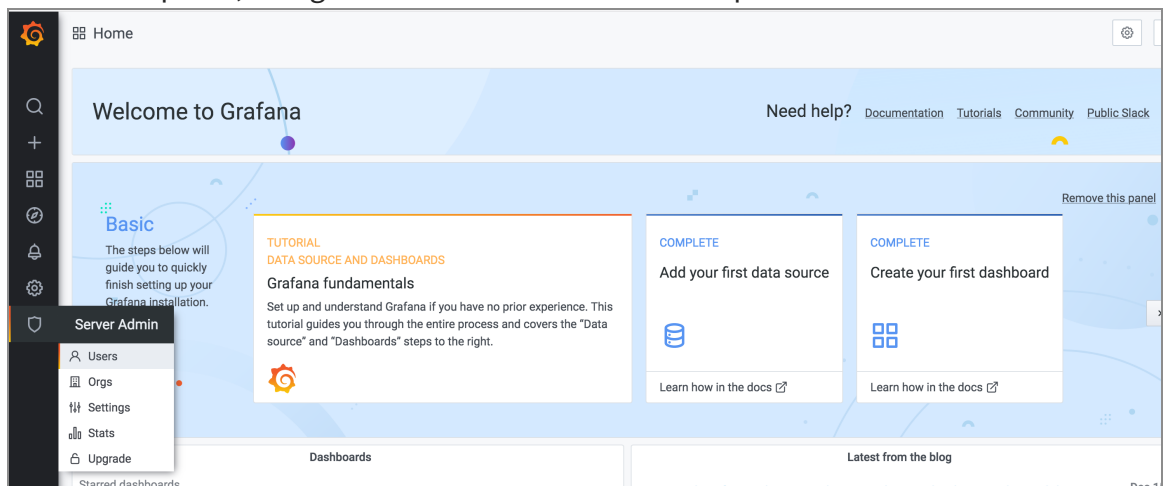
You can carry out operations using the latest credentials.

## Reporting

### Managing Users and Roles

The **masheryadmin** user can create users and assign roles according to their needs.

1. Login to Grafana dashboard as the **masheryadmin** user.
2. On the left panel, navigate to **Server Admin Shield** option.



3. Follow the Grafana documentation for instructions on [creating and adding a new user](#).
4. Assign **Roles (Permissions)** as described in the Grafana documentation on [Permissions](#).

## Port Configurations

The below are the port configurations available for deployment with options to configure https

Parameter	Port
tml_tm_http_port	80
tml_tm_https_port	443
tml_tm_mhttps_port	1443
tml_api_http_port	7080
tml_api_https_port	7443
tml_cm_http_port	8080
tml_cm_https_port	8443

## Securing Cassandra

Cassandra nodes communicate with each other via the Gossip protocol. In multi zone or multi region networks, it becomes necessary to communicate via SSL.

Enable SSL on Cassandra inter node communication.

### Enable encrypted inter-node communication in Cassandra cluster

Since Cassandra nodes communicate across regions via public network, it is highly recommended to encrypt inter-node communication in Cassandra cluster.

For example,

```
{
  "auto_binding":"ON",
  "dependency_health_check": "OFF",
  "server_internode_encryption": "all",
  "server_keystore_password": "changeme",
  "server_truststore_password": "changeme",
  "server_require_client_auth": true
}
```

## Securing MySQL

This sections describes how to change the default passwords for MySQL.

### Changing the MySQL Password of a Running Cluster

The following section describes how to change the MySQL password of a running cluster in Kubernetes.

If you have a running TIBCO Cloud™ API Management - Local Edition cluster in a Kubernetes environment, the cluster was created using the MySQL password specified in the `tml_cluster_properties.json` file.

To change the MySQL password:

#### Procedure

1. Change the passwords in the `tml_cluster_properties.json` file in your deployment folder.

For example, if you deployed from the `/jdoe/tml-530/docker-deploy/gcp/k8s/manifest-single-zone/` folder, modify the `tml_cluster_properties.json` file in that folder.

In this example, to make the new password `changeme12`, modify the following:

```
{
  "mysql_root_pwd": "changeme12",
  "mysql_masheryonprem_pwd": "changeme12",
  "mysql_mashonpremrepl_pwd": "changeme12",
  "mysql_mashclient_pwd": "changeme12",
```

```

    "mysql_masherybackup_pwd": "changeme12",
    "mom_server": "https://api-mom.mashery.com",
    "mom_key": "",
    "mom_secret": ""
  }

```

2. Delete the cluster-property Kubernetes secret:

```
kubectl delete secret cluster-property
```

3. Create the Kubernetes secret again using the `tml_cluster_properties.json` file containing the new password:

```
kubectl create secret generic cluster-property --from-file=./tml_
cluster_properties.json
```

4. Log in to the MySQL container:

```
kubectl exec -it mysql-set-0-0 /bin/bash
```

5. Connect to the MySQL server with a MySQL client using the old password, then set the new password.

```
mysql -u root -p
```

Execute the following commands on the MySQL prompt:

```

use masherysolar;
FLUSH PRIVILEGES;
ALTER USER 'root'@'localhost' IDENTIFIED BY 'changeme12';
ALTER USER 'masheryonprem'@'localhost' IDENTIFIED BY 'changeme12';
ALTER USER 'masheryonprem'@'%' IDENTIFIED BY 'changeme12';
ALTER USER 'mashonpremrepl'@'%' IDENTIFIED BY 'changeme12';
ALTER USER 'mashclient'@'%' IDENTIFIED BY 'changeme12';
ALTER USER 'masherybackup'@'%' IDENTIFIED BY 'changeme12';

```

6. Exit the MySQL container.
7. Delete the MySQL pod.

```
kubectl delete pod mysql-set-0-0
```

8. Once mysql-set-0-0 is created again, wait for it to become ACTIVE. Then delete all the cache pods; for example, if you have three cache pods, delete all as follows:

```
kubectl delete pod cache-set-0-0 cache-set-0-1 cache-set-0-2
```

9. Once cache pods are created again, delete all the tm pods; for example, if you have three tm pods, delete all as follows:

```
kubectl delete pod tm-deploy-0-764874c9d8-2gl5x tm-deploy-0-4c9d764878-2gl5x tm-deploy-0-747648c9d8-2gl5
```

10. Once the tm pods are created again, delete the cm pod as follows:

```
kubectl delete pod cm-deploy-0-7b9976697c-zblbk
```

11. Once all the pods are running, you can login to the cm pod and verify that the status is ACTIVE. Likewise, any existing service endpoint that was created before you applied the password changes should work as before.

## Encrypting MySQL Replication Group

The following section describes how to enable encryption for a MySQL replication group.

### Update properties/tml\_sql\_properties.json

Change the following property from default value "false" to "true":

```
"mysql_group_replication_encryption": true
```

### Update Key and Certificates

Keys and certificates can be created using either MySQL or openssl.

#### Creating SSL and RSA Certificates and Keys using MySQL



For more information on how to create SSL and RSA certificates and keys using MySQL, see [MySQL Reference Manual](#).

For example:

```
mysql_ssl_rsa_setup --datadir=${PWD}
```

This generates the following keys and certificates:

- ca-key.pem
- ca.pem
- client-cert.pem
- client-key.pem
- private\_key.pem
- public\_key.pem
- server-cert.pem
- server-key.pem

### Creating SSL Certificates and Keys using openssl

For more information on how to create SSL certificates and keys using openssl, see [MySQL Reference Manual](#).

### Overwrite the Example Keys and Certificates

Use the generated keys and certificates to overwrite the following files:

- properties/tml-sql-ca.pem
- properties/tml-sql-server-cert.pem
- properties/tml-sql-server-key.pem

## Securing Traffic

It is a good practice to use the HTTPS configuration. There are many ways in which traffic calls are secured, One way SSL, Mutual SSL, and so on.

# HTTPS Configuration Overview

## Non-mutual HTTPS

Message flow:

Client --(HTTPS 1)--> Customer Load Balancer --(HTTPS 2)--> Local Edition instance --(HTTPS 3)--> Backend Service

In the above flow:

1. HTTPS 1 is achieved between the Client and the Customer Load Balancer by appropriately configuring the Load Balancer. This is outside the scope of Local Edition.
2. HTTPS 2 configuration is what we refer to as the HTTPS Server feature. Since the Load Balancer and the Local Edition instance are in the customer's network, mutual SSL is currently not supported in the HTTPS server feature.
3. HTTPS 3 configuration is what we refer to as HTTPS Client feature. Since this call typically goes across networks, we support mutual SSL settings by configuring an HTTPS Client profile with Identity and Trust settings, and associating the profile with the endpoint configuration. The required configuration is documented in this section.

## Mutual HTTPS

Local Edition can be configured for mutual HTTPS authentication (server side). To accomplish this, you deploy a totally separated Local Edition cluster with mutual HTTPS authentication on it.

- In **tethered mode**, this totally separated Local Edition cluster syncs with a separated area in which all APIs to be protected by mutual HTTPS authentication are created.
- In **untethered mode**, you author all APIs to be protected by mutual HTTPS authentication using Configuration Manager. All APIs are confined in this separated Local Edition cluster.

## HTTPS Server Configuration

This following section describes the HTTPS Server Configuration.

## Traffic Manager as an HTTPS Server with Mutual SSL

Perform the following steps to set up Traffic Manager as an HTTPS server with mutual SSL.

### Procedure

1. Run the `upload_ssl_server_truststore_for_traffic_manager` Jenkins job in the Installer to upload the trust store.

Complete the fields in the Jenkins job as follows:

- `truststore_file` - The key store in PKCS#12 format, which holds all Certificate Authority (CA) certificates which are trusted.
- `truststore_password` - The password protecting the trust store.

The screenshot shows the Jenkins web interface for the job `upload_ssl_server_truststore_for_traffic_manager`. The left sidebar contains navigation links: Back to Dashboard, Status, Changes, Workspace, Build with Parameters, Delete Project, Configure, Rebuild Last, and Rename. The main area is titled 'Project upload\_ssl\_server\_truststore\_for\_traffic\_manager' and states 'This build requires parameters:'. There are two parameter fields: `truststore_file` with a 'Browse...' button and the text 'No file selected. SSL truststore file in PKCS#12 format.', and `truststore_password` with a masked password field and the text 'SSL truststore password'. A 'Build' button is located below these fields. At the bottom, there is a 'Build History' section showing a single build from Feb 14, 2020 11:40 PM, with links for 'Atom feed for all' and 'Atom feed for failures'.

The `upload_ssl_server_truststore_for_traffic_manager` Jenkins job uploads the trust store (`tml-tm-trust.jks`) to the `/var/jenkins_home/docker-deploy/properties` folder. This trust store holds all trusted CA certificates.



**Note:** There is a built-in sample, self-signed root CA certificate. You should upload your own trust store in the Local Edition installer

2. Configure the following property in the `/var/jenkins_home/docker-deploy/properties/tml_tm_properties.json` file:
  - `tml_truststore_password` - The password protecting the trust store.

For example:

```
"tm_truststore_password": "changeme",
```

3. Configure the following property in the `manifest-onprem-swarm.json` file:

- `tml_tm_mhttps_enabled` - Set to `true` to turn on mutual HTTPS authentication.

For example:

```
"tml_tm_mhttps_enabled": false,
```

4. Verify your mutual HTTPS authentication configuration using the following example `curl` command:

```
curl -k -v --key PATH_TO_KEY/yam_root_.pkcs8 --cert PATH_TO_CERT/yam_root_.cer:changeme -H 'host: calypsoqa.api.mashery.com' https://$LB:443/mock?api_key=mycustomkey
```

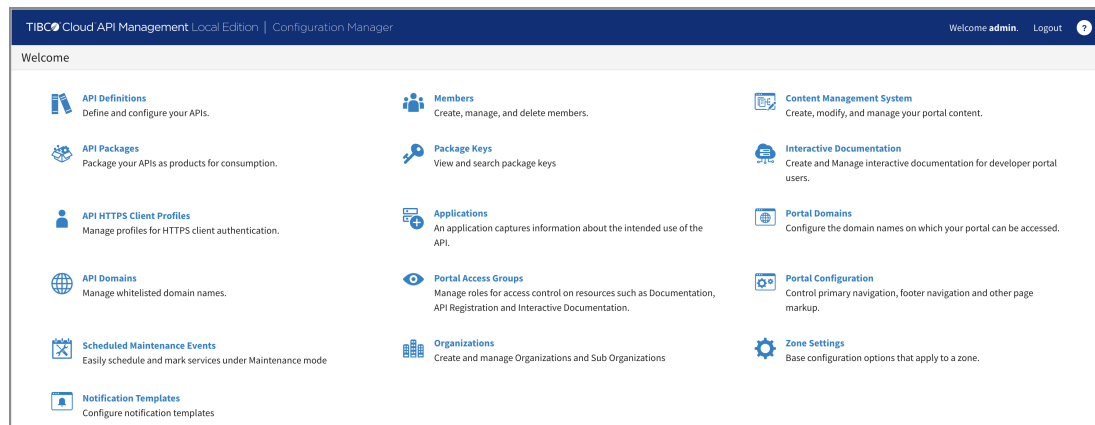
**i Note:** In `yam_root_.cer:changeme`, "`yam_root_.cer`" is the certificate file name, and "`changeme`" is the trust store password protecting the private key. "`LB`" is the public IP address of the Load Balancer for Traffic Manager.

## HTTPS Client Configuration

The following example walks through a use case where the CLI is used to manage the components within the cluster. The example also uses the (Untethered) Configuration Manager UI where necessary.

## HTTPS Endpoint Configuration

1. Set up an HTTPS Endpoint using the HTTPS Client Profile.
  - a. Open the Configuration Manager and sign in.



b. Under **API Definitions** open the **hw\_sample\_service** API for edit.

**API Definition** | Home / API Definitions / Hw\_sample\_service

**Security** | **Hw\_sample\_service** [Delete]

[Save] [Cancel]

**API Service Key**  
 uyhe3gmusp7kwjd3z6x75qbcm  
 Unique Identifier for an API Service, primarily used within Mashery's OAuth 2.0 feature. Sometimes referred to as an 'spkey', 'service\_key', 'Mashery API ID', or 'Service ID'.

**Organization**  
 Area Level

**Name**  
 hw\_sample\_service  
 API Name represents an API, or collection of endpoints, and associated settings. API Name will be referenced in creation of Packages/Plans, Reporting, etc. Name can be modified later.

**Description**  
 [Text Area]  
 Modify the description of the API. You can enter purpose, limitations, or other important information about the API.

**Version**  
 [Text Field]  
 Version identifier is meant to assist in the tracking of different versions of the API. This is a customer-managed property in that the version is not modified as property changes are made to the API definition.

**QPS Limit Overall**  
 0  
 Enter a throttle (max Queries per Second) value that limits the calls for this API.

**Cross Domain Policy**  
 <?xml version="1.0"?>  
 <!DOCTYPE cross-domain-policy SYSTEM  
 [Text Area]  
 Enter the data for cross-domain policy for API that can be accessed by using Flash.

**Robots Policy**  
 [Text Area]  
 Enter the data for robots policy. If no policy information is entered, the default policy allows bots an unrestricted access to the API.

c. Click on **Endpoints** from the left pane.

**API Definition** | Home / API Definitions / Hw\_sample\_service / Endpoints

**Endpoints** | my search terms [Search] [+ New]

| Name               | Updated           | URL                          | Actions         |
|--------------------|-------------------|------------------------------|-----------------|
| hw_sample_endpoint | Nov 17 2021 20:17 | http://api.example.com/hello | [Edit] [Delete] |

d. Click on the **hw\_sample\_endpoint** link in the panel.

- e. Select HTTPS protocol for the Origin Endpoint backend.

**Origin Endpoint**

|          |                          |        |                      |
|----------|--------------------------|--------|----------------------|
| Protocol | Domain(s)/IP Address(es) | Path   | Parameters To Append |
| HTTPS    | localhost:9080           | /hello |                      |

The above set of fields defines the Origin Endpoint, to which we will pass traffic based on the configured mapping between Inbound and Origin Endpoints. Please select the supported Protocol for this Endpoint in the Protocol dropdown, FQDN or IP Address in the Domain / IP Address field (e.g. api.wideworldofacme.com or 10.10.10.2), and Request path (e.g. /v1/user). You may optionally enter a set of static URI parameter values, which will be passed to your Origin Endpoint.

**HTTPS Client Profile**

one\_way\_profile\_001

Identity Store Name:

Trust Store Name: trust\_001

Allows you to specify an HTTPS Client Profile for use with the Origin Endpoint. Please see the documentation for more information.

## Testing the HTTPS Client Call

To test the HTTPS Client call, start testing traffic using CURL:

- From within the Cluster Manager CLI container:

```
[root@cm-deploy-0-76ffbdbb6b-jjb44 builder]# clustermanager ls
components
Using cluster [Local Edition Reference Cluster]
Using Zone [local]
Component ID                                Component Type
Component Name          Component Status      Component Host
Component Agent Port    Component Service Port(s)
-----
-----
0adf6ab7-19a3-4598-9c4e-f9e20729448f  sql                sql
ACTIVE                                10.244.1.5          9080
3306
bd465cb3-2c19-4b89-afc8-f4719cd43d7b  nosql              nosql
ACTIVE                                10.244.1.2          9080
9042
37e880ce-6d05-4a6f-bdae-cee24a70b6a0  cache              cache
ACTIVE                                10.244.1.6          9080
11212,11211,11213,11214,11215,11216
f8f6da07-6669-413c-b356-cdb26be4f4df  trafficmanager     tm
ACTIVE                                10.244.1.7          9080
8080
74062084-0d77-4a43-9551-07db90e8fdd9  logservice         log
ACTIVE                                10.244.1.4          9080
24224
```

```
[root@cm-deploy-0-76ffbdbb6b-jjb44 builder]# curl -H
'Host:api.example.com' 'http://x.x.x.x/hw_path_01?api_
key=sra663fmnshjazx5sv2mgtmw'
{"time":"2018-11-28 20:17:02.841+0000","message":"Hello world"}
```

## 2. From load balanced TM service:

**Note:** Note the Kubernetes service for the Traffic Manager ("tm-svc" in this case).

```
admin-MBP15:properties <admin>$ kubectl get svc
NAME                TYPE                CLUSTER-IP          EXTERNAL-IP
PORT(S)              AGE
cass-svc-0           ClusterIP           None                 <none>
9042/TCP              16h
kubernetes            ClusterIP           100.64.0.1          <none>
443/TCP              17h
log-svc-0            ClusterIP           None                 <none>
24224/TCP             16h
mysql-svc-0          ClusterIP           None                 <none>
3306/TCP              16h
tm-svc               LoadBalancer       100.71.130.12       a45e45c83f2cb...
80:32299/TCP,443:32117/TCP,8083:31296/TCP 16h

admin-MBP15:properties <admin>$ kubectl describe service tm-
svc|grep Ingress|awk -F' ' '{print $3}'
a45e45c83f2cb11e8bc330609f0835b9-95933830.us-east-
1.elb.amazonaws.com

admin-MBP15:properties <admin>$ curl -H 'Host:api.example.com'
'http://a45e45c83f2cb11e8bc330609f0835b9-95933830.us-east-
1.elb.amazonaws.com/hw_path_01?api_key=sra663fmnshjazx5sv2mgtmw' -k
{"time":"2018-11-28 21:15:04.540+0000","message":"Hello world"}
```

# Troubleshooting HTTPS Problems

## Enabling Java SSL Debug Logging

To enable Java SSL debug logging for Local Edition, follow the steps below:

## Procedure

1. Add the following setting in `/opt/javaproxy/proxy/proxy.ini` in the ml-tm container:

```
-Djavax.net.debug=all
```

2. Restart javaproxy:

```
service javaproxy restart
```

3. Send requests to Local Edition and watch the log in `/var/log/mashery/javaproxy-runtime.log`. For example:

```
tail -f /var/log/mashery/javaproxy-runtime.log
```

## HTTPS Mutual Authentication

The following sections describe how to set up and create HTTPS mutual authentication.

- [Setting up HTTPS Authentication](#)
- [Creating Mutual Authentication](#)

## Setting up HTTPS Authentication

HTTPS client profile is used to provide certificate/identity based authentication to endpoints. This can be done in two ways:

- [One-way HTTPS authentication](#)
- [Mutual authentication](#)

## One-Way HTTPS Client Configuration

To create a one-way HTTPS client configuration:

**i Note:** The following procedure is illustrated on a Kubernetes system.



## 1. Log in to the CLI container.

```
kubectlexec-it cm-deploy-0-76ffbdbb6b-jjb44envCOLUMNS=$COLUMNS
LINES=$LINES
TERM=$TERMbash
```

## 2. In the command line interface, input the scope of operation as below:

```
[root@cm-deploy-0-76ffbdbb6b-jjb44 builder]# clustermanager ls
clusters
Cluster ID                               Cluster Name
-----
6b08d0e3-637e-4e18-b628-2b934db0abfc   Local Edition

[root@cm-deploy-0-76ffbdbb6b-jjb44 builder]# clustermanager use
cluster 6b08d0e3-637e-4e18-b628-2b934db0abfc
Using cluster [Tibco Mashery Local Reference Cluster]
[root@cm-deploy-0-76ffbdbb6b-jjb44 builder]# clustermanager ls
zones
Using cluster [Local Edition Reference Cluster]
Zone ID                               Zone Name
-----
0f7c76d8-feb6-4d65-8dd9-532e43fe7eef   local
[root@cm-deploy-0-76ffbdbb6b-jjb44 builder]# clustermanager use
zone 0f7c76d8-feb6-4d65-8dd9-532e43fe7eef
Using cluster [Local Edition Reference Cluster]
Using Zone [local]
```

## 3. Upload the trust certificate.

## a. Copy the trust certificate file to the CLI container.

```
admin-MBP15:properties admin$ kubectl get pods
NAME                                READY    STATUS    RESTARTS
AGE
cache-deploy-0-577f7d5c7b-42pfn    1/1      Running   0
47m
cass-set-0-0                        1/1      Running   0
48m
cm-deploy-0-76ffbdbb6b-jjb44        1/1      Running   0
47m
log-set-0-0                        1/1      Running   0
47m
mysql-set-0-0                      1/1      Running   0
```

```

47m
tm-deploy-0-5b584758bb-9m6cn      1/1      Running    0
47m

admin-MBP15:properties admin$ kubectl cp
~/Downloads/samplecert.cer cm-deploy-0-76ffbdbb6b-
jjb44:/usr/local/bin/

```

- b. Upload the trust certificate as shown:

```

[root@cm-deploy-0-76ffbdbb6b-jjb44 builder]# clustermanager
create certificate --file /usr/local/bin/samplecert.cer
Creating certificate for cluster ID 6b08d0e3-637e-4e18-b628-
2b934db0abfc and zone ID 0f7c76d8-feb6-4d65-8dd9-532e43fe7eef
Successfully created certificate for cluster ID 6b08d0e3-637e-
4e18-b628-2b934db0abfc of zone ID 0f7c76d8-feb6-4d65-8dd9-
532e43fe7eef

```

- c. For tethered mode, update the trust as shown:

```

[root@cm-deploy-0-76ffbdbb6b-jjb44 builder]# clustermanager
set certificate --uuid cb4d2bc2-f863-440b-1001-1d11369404d8 --
file /usr/local/bin/samplecert.cer
Updating certificate for cluster ID 6b08d0e3-637e-4e18-b628-
2b934db0abfc and zone ID 0f7c76d8-feb6-4d65-8dd9-532e43fe7eef
Successfully updated certificate for cluster ID 6b08d0e3-637e-
4e18-b628-2b934db0abfc of zone ID 0f7c76d8-feb6-4d65-8dd9-
532e43fe7eef

```

4. Open the Configuration Manager and sign in. Create the new configuration.

TIBCO Cloud API Management Local Edition | Configuration Manager

Welcome admin. Logout ?

### API HTTPS Client Profile

Save Cancel

**Name**

One-way

☐ Verify Hostname

**API HTTPS Client Identity**

None

**Choose API HTTPS Client Trusts**

✓ Acme.Client.Example.Com

## Mutual HTTPS Client Configuration

localhost:5000/api-definitions/56112/endpoints/178725/edit

TIBCO Cloud API Management Local Edition | Configuration Manager

CORS

Call Transformation

More

hw\_sample\_endpoint

Endpoint Name. Used within Packages/Plans and Call Inspector.

**Published Endpoint**

Protocol Domain Path

HTTPS api.example.com /hw\_path\_01

The above set of fields, along with the selected HTTP Methods (found on the Protocol & Authentication tab) define the Published Endpoint. Please select the supported Protocol for this Endpoint in the Protocol dropdown, FQDN in the Domain field (e.g. api.wideworldofacme.com), and Request path (e.g. /v1/user).

**Origin Endpoint**

Protocol Domain(s)/IP Address(es) Path Parameters To Append

HTTPS localhost:9080 /hello

The above set of fields defines the Origin Endpoint, to which we will pass traffic based on the configured mapping between Inbound and Origin Endpoints. Please select the supported Protocol for this Endpoint in the Protocol dropdown, FQDN or IP Address in the Domain / IP Address field (e.g. api.wideworldofacme.com or 10.10.10.2), and Request path (e.g. /v1/user). You may optionally enter a set of static URI parameter values, which will be passed to your Origin Endpoint.

**HTTPS Client Profile**

mutual\_auth\_profile\_001

Allows you to specify an HTTPS Client Profile for use with the Origin Endpoint. Please see the documentation for more information.

**Identity Store Name:**  
identity\_001

**Trust Store Name:**  
trust\_021

In addition to the previous steps mentioned for one-way HTTP Client Configuration profiles, do the following:

1. List all the existing identities:

```
[root@cm-deploy-0-7d69cb94f7-g4gj9 builder]# clustermanager ls
identities
```

```
clusterId [6bb97326-007d-4ba3-828e-24bb81449b94] and zoneId
```

```
[5d75a537-a6b4-489c-8d5a-d97469a7390a]
```

```
Using cluster name [Raj Cluster]
```

```
Using Zone name [us-east-1e]
```

| UUID                                 | Area ID | Expiration | Name         |
|--------------------------------------|---------|------------|--------------|
| cb4d2bc2-f863-440b-0001-1d11369404d8 | 5488    |            | identity_001 |
| cb4d2bc2-f863-440b-0002-1d11369404d8 | 5488    |            | identity_002 |
| cb4d2bc2-f863-440b-0003-1d11369404d8 | 5488    |            | identity_003 |
| cb4d2bc2-f863-440b-0004-1d11369404d8 | 5488    |            | identity_004 |
| cb4d2bc2-f863-440b-0005-1d11369404d8 | 5488    |            | identity_005 |
| cb4d2bc2-f863-440b-0006-1d11369404d8 | 5488    |            | identity_006 |
| cb4d2bc2-f863-440b-0007-1d11369404d8 | 5488    |            | identity_007 |
| cb4d2bc2-f863-440b-0008-1d11369404d8 | 5488    |            | identity_008 |
| cb4d2bc2-f863-440b-0009-1d11369404d8 | 5488    |            | identity_009 |
| cb4d2bc2-f863-440b-0010-1d11369404d8 | 5488    |            | identity_010 |
| cb4d2bc2-f863-440b-0011-1d11369404d8 | 5488    |            | identity_011 |
| cb4d2bc2-f863-440b-0012-1d11369404d8 | 5488    |            | identity_012 |
| cb4d2bc2-f863-440b-0013-1d11369404d8 | 5488    |            | identity_013 |
| cb4d2bc2-f863-440b-0014-1d11369404d8 | 5488    |            | identity_014 |
| cb4d2bc2-f863-440b-0015-1d11369404d8 | 5488    |            | identity_015 |
| cb4d2bc2-f863-440b-0016-1d11369404d8 | 5488    |            | identity_016 |
| cb4d2bc2-f863-440b-0017-1d11369404d8 | 5488    |            | identity_017 |

```
cb4d2bc2-f863-440b-0018-1d11369404d8    identity_018
5488
cb4d2bc2-f863-440b-0019-1d11369404d8    identity_019
5488
cb4d2bc2-f863-440b-0020-1d11369404d8    identity_020
5488
```

2. Copy the Identity file to the CLI container.

```
kubectll cp ~/Downloads/sample-identity.p12 cm-deploy-0-7d69cb94f7-
g4gj9:/usr/local/bin/
```

3. Use the CLI command to update the identity:

```
[root@cm-deploy-0-7d69cb94f7-g4gj9 builder]# clustermanager ls
identities
clusterId [6bb97326-007d-4ba3-828e-24bb81449b94] and zoneId
[5d75a537-a6b4-489c-8d5a-d97469a7390a]
Using cluster name [Raj Cluster]
Using Zone name [us-east-1e]
UUID                                     Name
      Area ID      Expiration
-----
cb4d2bc2-f863-440b-0001-1d11369404d8    identity_001
5488
cb4d2bc2-f863-440b-0002-1d11369404d8    identity_002
5488
cb4d2bc2-f863-440b-0003-1d11369404d8    identity_003
5488
cb4d2bc2-f863-440b-0004-1d11369404d8    identity_004
5488
cb4d2bc2-f863-440b-0005-1d11369404d8    identity_005
5488
cb4d2bc2-f863-440b-0006-1d11369404d8    identity_006
5488
cb4d2bc2-f863-440b-0007-1d11369404d8    identity_007
5488
cb4d2bc2-f863-440b-0008-1d11369404d8    identity_008
5488
cb4d2bc2-f863-440b-0009-1d11369404d8    identity_009
5488
cb4d2bc2-f863-440b-0010-1d11369404d8    identity_010
5488
```

|  |              |
|--|--------------|
| cb4d2bc2-f863-440b-0011-1d11369404d8<br>5488 | identity_011 |
| cb4d2bc2-f863-440b-0012-1d11369404d8<br>5488 | identity_012 |
| cb4d2bc2-f863-440b-0013-1d11369404d8<br>5488 | identity_013 |
| cb4d2bc2-f863-440b-0014-1d11369404d8<br>5488 | identity_014 |
| cb4d2bc2-f863-440b-0015-1d11369404d8<br>5488 | identity_015 |
| cb4d2bc2-f863-440b-0016-1d11369404d8<br>5488 | identity_016 |
| cb4d2bc2-f863-440b-0017-1d11369404d8<br>5488 | identity_017 |
| cb4d2bc2-f863-440b-0018-1d11369404d8<br>5488 | identity_018 |
| cb4d2bc2-f863-440b-0019-1d11369404d8<br>5488 | identity_019 |
| cb4d2bc2-f863-440b-0020-1d11369404d8<br>5488 | identity_020 |

To create a new identity, see [Create Identity](#).

To update an existing identity, see [Update Identity](#).

## Cluster Manager CLI Commands

- [Create Certificate](#)
- [Create Identity](#)
- [Update Certificate](#)
- [Update Identity](#)
- [List Certificates](#)
- [List Identities](#)

## Create Certificate

Example:

```
[root@01294de80b3d builder]# clustermanager create certificate help
Error: required flag(s) "file" not set
Usage:
clustermanager create certificate [flags]

Aliases:
certificate, cert

Flags:
--file string certificate file to create, CER type
-h, --help help for certificate

required flag(s) "file" not set

[root@01294de80b3d builder]# clustermanager create cert --file
sample.cer
Creating a certificate for the topology
Successfully created certificate for the topology
```

## Create Identity

Example:

```
[root@01294de80b3d builder]# clustermanager create identity help
Error: required flag(s) "file", "password" not set
Usage:
clustermanager create identity [flags]

Aliases:
identity, idty

Flags:
--file string          identity file to create, P12 / JKS type
-h, --help             help for identity
--password string      password for the identity file

required flag(s) "file", "password" not set

[root@01294de80b3d builder]# clustermanager create identity --file
sample.p12 --password password
```

Creating an identity for the topology  
 Successfully created identity for the topology

## Update Certificate

Example:

```
[root@01294de80b3d builder]# clustermanager set certificate help
Error: required flag(s) "file", "uuid" not set
Usage:
  clustermanager set certificate [flags]

Aliases:
  certificate, cert

Flags:
  --file string      certificate file to upload, CER type
  -h, --help         help for certificate
  --uuid string      unique ID for the existing certificate that
                     needs update

required flag(s) "file", "uuid" not set

[root@01294de80b3d builder]# clustermanager set cert --file sample.cer -
--uuid f26a2422-b2cd-4905-bbc7-5706d450b137
Updating certificate for the topology
Successfully updated certificate for the topology
```

## Update Identity

Example:

```
[root@01294de80b3d builder]# clustermanager set identity help
Error: required flag(s) "file", "password", "uuid" not set
Usage:
  clustermanager set identity [flags]

Aliases:
  identity, idty

Flags:
```



```

    --file string      identity file to upload, P12 / JKS type
    -h, --help         help for identity
    --password string  password for the identity file
    --uuid string      unique ID for the existing identity that
needs update
    --verifyHostname  required for mutual SSL. If set to true,
hostname inside client and server identity & trust files should be same.
'verifyHostname=false' if the flag is empty

required flag(s) "file", "password", "uuid" not set

```

```

[root@01294de80b3d builder]# clustermanager set idty --file sample.p12 -
--password password --uuid a434bb96-be67-4676-a8aa-9c9fe6503d76
Updating an identity for the topology
Successfully updated identity for the topology

```

## List Certificates

Example:

```

[root@01294de80b3d builder]# clustermanager list certificates --help
List of trust certificates available for the topology

```

```

Usage:
  clustermanager list certificates [flags]

```

```

Flags:
  -h, --help          help for certificates

```

```

[root@01294de80b3d builder]# clustermanager ls certificates

```

| UUID                                 | Area ID | Expiration | Name   |
|--------------------------------------|---------|------------|--------|
| f26a2422-b2cd-4905-bbc7-5706d450b137 | 0       | 2029-03-22 | sample |

## List Identities

### Example

```
[root@01294de80b3d builder]# clustermanager list identities --help
List of identities available for the topology
```

#### Usage:

```
clustermanager list identities [flags]
```

#### Flags:

```
-h, --help            help for identities
```

```
[root@01294de80b3d builder]# clustermanager list identities
```

```
Using Zone name [local]
```

| UUID                                 | Area ID | Expiration | Name   |
|--------------------------------------|---------|------------|--------|
| a434bb96-be67-4676-a8aa-9c9fe6503d76 | 5488    | 2020-03-26 | sample |

## Securing Using OAuth

Endpoints can be configured as protected or unprotected i.e without any authentication. TIBCO Cloud™ API Management - Local Edition provides authentication using the OAuth method.

It is advised to check the recommendations for OAuth 5.x

## OAuth APIs and Authenticator Service

The OAuth Authenticator introduced in Local Edition 5.0 includes the Basic Authenticator for an OAuth API endpoint, and a Public Key Authenticator configured and targeted for an OAuth service endpoint. This page describes the configuration of the OAuth Authenticator and also provides sample calls to OAuth APIs.

### OAuth Authenticator Service Configuration

Authorization for the OAuth Authenticator Service depends on the configuration of "com.mashery.service.onprem.oauth.authenticator.oauth-service-authenticator" in `tml_tm_`

properties.json.

**Note:** For more information on the tml\_tm\_properties.json file, refer to the *tml-tm* section in the [Configuring Properties Common to All Deployments](#) topic.

OAuth API requests can be authenticated using Basic Authentication and/or Public Key Authentication:

- If the OAuth API endpoint is exposed or used directly, Basic Authentication should be configured.
- If a proxy OAuth API service endpoint is used, Public Key Authentication should be configured.

The following is an example of the configuration in tml\_tm\_properties.json for both Basic and Public Key Authentication:

```
"com.mashery.service.onprem.oauth.authenticator.oauth-service-
authenticator" : {
    "publicKeyName": "public_key",
    "publicKeyValue": "3bv8u3p8l",
    "username": "root",
    "password":
"057ba03d6c44104863dc7361fe4578965d1887360f90a0895882e58a6248fc86"
},
```

- The username/password pair is to support Basic Authentication. The password can be generated by any SHA256 Hash generator. "057ba03d6c44104863dc7361fe4578965d1887360f90a0895882e58a6248fc86" is a SHA256 for "changeme".

The encoded string "cm9vdDpjGfU2VtZQ==" in the curl header -H 'Authorization: Basic cm9vdDpjGfU2VtZQ==' is for "root:changeme".

- The publicKeyName/publicKeyValue pair is for public key authentication when setting up an OAuth API service endpoint.

The public key is extracted from the request parameters when the request is received by the OAuth API service endpoint created in either tethered or untethered mode (refer to sample OAuth API service endpoint setup below for more details).

To use the endpoint, the publicKeyName and publicKeyValue properties must be configured in tml\_tm\_properties.json and their values should match their corresponding parameter string in the endpoint.

For example, if `publicKeyName/publicKeyValue` is set to `"public_key/3bvu3p8l "`, its parameter string should be `"public_key=3bvu8u3p8l"`.

- Basic Authenticator is disabled if the `username` property of service `"com.mashery.service.onprem.oauth.authenticator.oauth-service-authenticator"` is not present in `tml_tm_properties.json`.

The request must pass public key authentication when Basic Authentication is disabled.

- Public key authentication is disabled if the `publicKeyName` property of service `"com.mashery.service.onprem.oauth.authenticator.oauth-service-authenticator"` is not present in `tml_tm_properties.json`.

If public key authentication is disabled, Basic Authentication should be enabled by configuring the username and password.

## Different Ways to Make OAuth API Calls

### Call via Internal OAuth API Endpoint Basic Authentication

```
curl -v -d '{"id":1,"method":"oauth2.createAccessToken","params":
{"service_key":"m9f5kyzfjjrssz6d5sfkbuw3","client":
{"client_id":"yq7chp6ufkeea587gp6wqa6x","client_
secret":"Azn4Xa"},"token_data":
{"grant_type":"implicit"},"uri":
{"redirect_uri":"https://some.com/cb"},"user_
context":"testUser1455730448397"},"jsonrpc":"2.0"}'
'https://api.mashery.com:8083/v2/json-rpc' -u root:changeme -k
```

This call uses the username/password pair in `tml_tm_properties.json` and is for API Management - Local Edition 4.x backward compatibility.

### Call via Proxied OAuth API Endpoint with Public Key Authentication

```
curl -v -d '{"id":1,"method":"oauth2.createAccessToken","params":
{"service_key":"m9f5kyzfjjrssz6d5sfkbuw3","client":
{"client_id":"yq7chp6ufkeea587gp6wqa6x","client_
secret":"Azn4Xa"},"token_data":
{"grant_type":"implicit"},"uri":
{"redirect_uri":"https://some.com/cb"},"user_
context":"testUser1455730448397"},"jsonrpc":"2.0"}' -H 'Content-Type:
application/json' 'http://api.mashery.com/oauth/authorization/?api_
key=8qavf83pg2uzgndtxz2w4dg3'
```

This call is new in Local Edition 5.x. It uses the URL path `/oauth/authorization` and `publicKeyName/publicKeyValue` pair in `tml_tm_properties.json`, along with the URL parameter string setting in the proxy endpoint.

Users do not need to specify any credentials (username/password) in the curl commands.

### Sample Proxy Endpoint Setup with `publicKeyName/publicKeyValue` Parameter String in Tethered/Untethered

The following screenshot shows the settings on "Call Transformation" of the proxied OAuth API endpoint.

#### Published Endpoint

| Protocol   | Domain          | Path                 |
|------------|-----------------|----------------------|
| HTTP/HTTPS | api.mashery.com | /oauth/authorization |

The above set of fields, along with the selected HTTP Methods (found on the Protocol & Authentication tab) define the Published Endpoint. Please select the supported Protocol for this Endpoint in the Protocol dropdown, FQDN in the Domain field (e.g. api.wideworldofacme.com), and Request path (e.g. /v1/user).

#### Origin Endpoint

| Protocol | Domain(s)/IP Address(es) | Path         | Parameters To Append |
|----------|--------------------------|--------------|----------------------|
| HTTPS    | localhost:8083           | /v2/json-rpc | public_key=3bv8u3p8l |

The above set of fields defines the Origin Endpoint, to which we will pass traffic based on the configured mapping between Inbound and Origin Endpoints. Please select the supported Protocol for this Endpoint in the Protocol dropdown, FQDN or IP Address in the Domain / IP Address field (e.g. api.wideworldofacme.com or 10.10.10.2), and Request path (e.g. /v1/user). You may optionally enter a set of static URI parameter values, which will be passed to

In this setup:

- The service URL is set to `/oauth/authorization` and can either be sent with HTTP or HTTPS.
- The parameters include the `publicKeyName/publicKeyValue` set in `tml_tm_properties.json`
- The protocol and IP address of the Origin Endpoint correspond to the internal OAuth API endpoint setting in `tml_tm_properties.json`.

## Support for HTTPS Connection

This setting is in `tml_tm_properties.json` to support an HTTPS connection for OAuth.

```
"com.mashery.service.onprem.oauth.api-server": {
  "http.enabled": true,
  "https.enabled": true,
  "http.port": 9083,
  "https.port": 8083,
  "ssl.keystore": "/etc/mashery-server-ssl/tml-tm.jks",
  "ssl.password": "changeit",
  "ssl.keypassword": "changeit"
},
```

In Local Edition 5.x, only `https.port` is configured in the `.yml` file and exposed when the container is started. The `http.port` works only in calls within the containers.

You should always use HTTPS configuration for OAuth API calls outside the container.

In the "properties" folder in Local Edition deployment scripts, "tml-tm.jks" should be replaced with customers' keystore. This keystore is used by both regular traffic (non-OAuth requests) and OAuth API.

After "tml-tm.jks" is replaced with customers' keystore, the setting "tm\_keystore\_password" in "properties/tml\_tm\_properties.json" should also be updated.

## OAuth APIs

Sample calls through proxy endpoint using above settings.

### Create access token

```
curl -v -d '{"id":1,"method":"oauth2.createAccessToken","params":
{"service_key":"m9f5kyzfjjrssz6d5sfkbuw3","client":{"client_
id":"yq7chp6ufkeea587gp6wqa6x","client_secret":"Azn4Xa"},"token_data":
{"grant_type":"implicit"},"uri":{"redirect_
uri":"https://some.com/cb"},"user_
context":"testUser1455730448397"},"jsonrpc":"2.0"}' -H 'Host:
api.mashery.com' -H 'Content-Type: application/json'
'http://X.X.99.100:80/oauth/authorization/?api_
key=8qavf83pg2uzgndtxz2w4dg3'
* Trying X.X.99.100...
* Connected to X.X.99.100 (X.X.99.100) port 80 (#0)
> POST /oauth/auth/v2/?api_key=8qavf83pg2uzgndtxz2w4dg3 HTTP/1.1
> Host: api.mashery.com
```

```
> User-Agent: curl/7.43.0
> Accept: */*
> Content-Type: application/json
> Content-Length: 310
>
* upload completely sent off: 310 out of 310 bytes
< HTTP/1.1 200 OK
< X-Mashery-Responder: 8abfe284898c
< Content-Type: application/json; charset=UTF-8
< Cache-Control: no-store
< Content-Length: 231
< Date: Tue, 30 Oct 2018 00:38:26 GMT
<
* Connection #0 to host X.X.99.100 left intact
{"jsonrpc": "2.0", "id": 1, "result": {"token_type": "bearer", "return_type": "json", "access_token": "unrct8sweny8h3b3kw39wab", "expires_in": 36000, "scope": null, "user_context": "testUser1455730448397", "uri": null, "extended": null, "state": null}}
```

## Create authorization code

```
curl -v -d '{ "id":1,"method":"oauth2.createAuthorizationCode","params":
{"service_key":"m9f5kyzfjjrssz6d5sfkbuw3","client":{"client_
id":"yq7chp6ufkeea587gp6wqa6x"},"response_type":"code","uri":{"redirect_
uri":"https://some.com/cb"},"user_context":
"testUser1455730448397"},"jsonrpc":"2.0"}' -H 'Host: api.mashery.com' -H
'Content-Type: application/json'
'http://X.X.99.100:80/oauth/authorization/?api_
key=8qavf83pg2uzgndtxz2w4dg3'
* Trying X.X.99.100...
* Connected to X.X.99.100 (X.X.99.100) port 80 (#0)
> POST /oauth/auth/v2/?api_key=8qavf83pg2uzgndtxz2w4dg3 HTTP/1.1
> Host: api.mashery.com
> User-Agent: curl/7.43.0
> Accept: */*
> Content-Type: application/json
> Content-Length: 277
>
* upload completely sent off: 277 out of 277 bytes
< HTTP/1.1 200 OK
< X-Mashery-Responder: 8abfe284898c
< Content-Type: application/json; charset=UTF-8
< Cache-Control: no-store
< Content-Length: 168
< Date: Tue, 30 Oct 2018 00:42:12 GMT
```

```
<
* Connection #0 to host X.X.99.100 left intact
{"jsonrpc":"2.0","id":1,"result":
{"code":"p2kjsmn8mw2xb3d2ksryxk6z","uri":{"redirect_
uri":"https://some.com/cb?code=p2kjsmn8mw2xb3d2ksryxk6z","state":""}},"e
rror":null}
```

## Create access token with authorization code

```
curl -v -d '{"id":1,"method":"oauth2.createAccessToken","params":
{"service_key":"m9f5kyzfjjrssz6d5sfkbuw3","client":{"client_
id":"yq7chp6ufkeea587gp6wqa6x","client_secret":"Azn4Xa"},"token_data":
{"grant_type":"authorization_
code","code":"p2kjsmn8mw2xb3d2ksryxk6z","scope":"scope1"},"uri":
{"redirect_uri":"https://some.com/cb"},"user_context":
"testUser1455730448397"},"jsonrpc":"2.0"}' -H 'Content-Type:
application/json' -H 'Host: api.mashery.com'
'http://X.X.99.100:80/oauth/authorization/?api_
key=8qavf83pg2uzgndtxz2w4dg3'
* Trying X.X.99.100...
* Connected to X.X.99.100 (X.X.99.100) port 80 (#0)
> POST /oauth/auth/v2/?api_key=8qavf83pg2uzgndtxz2w4dg3 HTTP/1.1
> Host: api.mashery.com
> User-Agent: curl/7.43.0
> Accept: */*
> Content-Type: application/json
> Content-Length: 372
>
* upload completely sent off: 372 out of 372 bytes
< HTTP/1.1 200 OK
< X-Mashery-Responder: 8abfe284898c
< Content-Type: application/json; charset=UTF-8
< Cache-Control: no-store
< Content-Length: 274
< Date: Tue, 30 Oct 2018 00:49:18 GMT
<
* Connection #0 to host X.X.99.100 left intact
{"jsonrpc":"2.0","id":1,"result":{"token_type":"bearer","return_
type":"json","access_token":"rjzb2k26cgv7tufe2j8cs7rm","expires_
in":36000,"refresh_token":"zgwyg64s2ejc6catffvg5kbx","scope":null,"user_
context":"testUser1455730448397","uri":null,"extended":null,"state":nul
l}}
```



## Update access token

```
curl -v -d '{"id":1,"method":"oauth2.updateAccessToken","params":
{"service_key":"m9f5kyzfjjrssz6d5sfkbuw3","client":{"client_
id":"yq7chp6ufkeea587gp6wqa6x"},"client_secret":"Azn4Xa"},"access_
token":"rjzb2k26cgv7tufe2j8cs7rm","user_
context":"testUser1455730448397", "expires_in":600},"jsonrpc":"2.0"}' -H
'Host: api.mashery.com' -H 'Content-Type: application/json'
'http://X.X.99.100:80/oauth/authorization/?api_
key=8qavf83pg2uzgndtxz2w4dg3'
* Trying X.X.99.100...
* Connected to X.X.99.100 (X.X.99.100) port 80 (#0)
> POST /oauth/auth/v2/?api_key=8qavf83pg2uzgndtxz2w4dg3 HTTP/1.1
> Host: api.mashery.com
> User-Agent: curl/7.43.0
> Accept: */*
> Content-Type: application/json
> Content-Length: 287
>
* upload completely sent off: 287 out of 287 bytes
< HTTP/1.1 200 OK
< X-Mashery-Responder: 8abfe284898c
< Content-Type: application/json; charset=UTF-8
< Cache-Control: no-store
< Content-Length: 322
< Date: Tue, 30 Oct 2018 01:01:50 GMT
<
* Connection #0 to host X.X.99.100 left intact
{"jsonrpc":"2.0","id":1,"result":{"token_type":"bearer","access_
token":"rjzb2k26cgv7tufe2j8cs7rm","expires":"2018-10-
30T11:01:50Z","refresh_token_expires":1540864158505,"scope":null,"user_
context":"testUser1455730448397","uri":null,"grant_type":"authorization_
code","client_id":"yq7chp6ufkeea587gp6wqa6x","extended":null}}
```

## Fetch access token

```
curl -v -d '{"id": 1,"jsonrpc":"2.0","method":
"oauth2.fetchAccessToken","params": {"service_
key":"m9f5kyzfjjrssz6d5sfkbuw3","access_token":
"rjzb2k26cgv7tufe2j8cs7rm"}}' -H 'Host: api.mashery.com' -H 'Content-
Type: application/json' 'http://X.X.99.100:80/oauth/authorization/?api_
key=8qavf83pg2uzgndtxz2w4dg3'
* Trying X.X.99.100...
* Connected to X.X.99.100 (X.X.99.100) port 80 (#0)
> POST /oauth/auth/v2/?api_key=8qavf83pg2uzgndtxz2w4dg3 HTTP/1.1
> Host: api.mashery.com
```

```
> User-Agent: curl/7.43.0
> Accept: */*
> Content-Type: application/json
> Content-Length: 157
>
* upload completely sent off: 157 out of 157 bytes
< HTTP/1.1 200 OK
< X-Mashery-Responder: 8abfe284898c
< Content-Type: application/json; charset=UTF-8
< Cache-Control: no-store
< Content-Length: 242
< Date: Tue, 30 Oct 2018 01:05:25 GMT
<
* Connection #0 to host X.X.99.100 left intact
{"jsonrpc": "2.0", "id": 1, "result": {"token_type": "bearer", "expires": "2018-10-30T11:01:50Z", "scope": null, "user_context": "testUser1455730448397", "uri": null, "grant_type": "authorization_code", "client_id": "yq7chp6ufkeea587gp6wqa6x", "extended": null}}
```

## Fetch application

```
curl -v -d '{"id":
1,"jsonrpc":"2.0","method":"oauth2.fetchApplication","params":{"service_
key":"m9f5kyzfjjrssz6d5sfkbuw3","client":{"client_
id":"yq7chp6ufkeea587gp6wqa6x","client_secret":"Azn4Xa"}}}' -H 'Host:
api.mashery.com' -H 'Content-Type: application/json'
'http://X.X.99.100:80/oauth/authorization/?api_
key=8qavf83pg2uzgndtxz2w4dg3'
* Trying X.X.99.100...
* Connected to X.X.99.100 (X.X.99.100) port 80 (#0)
> POST /oauth/auth/v2/?api_key=8qavf83pg2uzgndtxz2w4dg3 HTTP/1.1
> Host: api.mashery.com
> User-Agent: curl/7.43.0
> Accept: */*
> Content-Type: application/json
> Content-Length: 187
>
* upload completely sent off: 187 out of 187 bytes
< HTTP/1.1 200 OK
< X-Mashery-Responder: 8abfe284898c
< Content-Type: application/json; charset=UTF-8
< Cache-Control: no-store
< Content-Length: 82
< Date: Tue, 30 Oct 2018 01:10:58 GMT
<
* Connection #0 to host X.X.99.100 left intact
```

```
{ "jsonrpc": "2.0", "id": 1, "result":
  { "id": 251547, "name": "Test0AuthRegisterCallBack" }}
```

## Fetch user application

```
curl -v -d '{ "id": 1, "method": "oauth2.fetchUserApplications",
  "params": { "service_key": "m9f5kyzfjjrssz6d5sfkbuw3", "user_
context": "testUser1455730448397" }, "jsonrpc": "2.0" }' -H 'Host:
api.mashery.com' -H 'Content-Type: application/json'
'http://X.X.99.100:80/oauth/authorization/?api_
key=8qavf83pg2uzgndtxz2w4dg3'
* Trying X.X.99.100...
* Connected to X.X.99.100 (X.X.99.100) port 80 (#0)
> POST /oauth/auth/v2/?api_key=8qavf83pg2uzgndtxz2w4dg3 HTTP/1.1
> Host: api.mashery.com
> User-Agent: curl/7.43.0
> Accept: */*
> Content-Type: application/json
> Content-Length: 162
>
* upload completely sent off: 162 out of 162 bytes
< HTTP/1.1 200 OK
< X-Mashery-Responder: 8abfe284898c
< Content-Type: application/json; charset=UTF-8
< Cache-Control: no-store
< Content-Length: 195
< Date: Tue, 30 Oct 2018 01:12:32 GMT
<
* Connection #0 to host X.X.99.100 left intact
{"jsonrpc": "2.0", "id": 1, "result":
  [{ "id": 251547, "name": "Test0AuthRegisterCallBack", "access_tokens":
    [ "rjzb2k26cgv7tufe2j8cs7rm", "unrct8sweny8h3b3kw39wab" ], "client_
id": "yq7chp6ufkeea587gp6wqa6x" ]}]}
```

## Revoke access token

```
curl -v -d '{ "id": 1, "method": "oauth2.revokeAccessToken", "params": {
  "service_key": "m9f5kyzfjjrssz6d5sfkbuw3", "client": { "client_
id": "yq7chp6ufkeea587gp6wqa6x", "client_secret": "Azn4Xa" }, "access_
token": "rjzb2k26cgv7tufe2j8cs7rm" }, "jsonrpc": "2.0" }' -H 'Host:
api.mashery.com' -H 'Content-Type: application/json'
'http://X.X.99.100:80/oauth/authorization/?api_
key=8qavf83pg2uzgndtxz2w4dg3'
* Trying X.X.99.100...
* Connected to X.X.99.100 (X.X.99.100) port 80 (#0)
```

```

> POST /oauth/auth/v2/?api_key=8qavf83pg2uzgndtxz2w4dg3 HTTP/1.1
> Host: api.mashery.com
> User-Agent: curl/7.43.0
> Accept: */*
> Content-Type: application/json
> Content-Length: 240
>
* upload completely sent off: 240 out of 240 bytes
< HTTP/1.1 200 OK
< X-Mashery-Responder: 8abfe284898c
< Content-Type: application/json; charset=UTF-8
< Cache-Control: no-store
< Content-Length: 38
< Date: Tue, 30 Oct 2018 01:17:10 GMT
<
* Connection #0 to host X.X.99.100 left intact
{"jsonrpc":"2.0","id":1,"result":true}

```

## Revoke user application

```

curl -v -d '{ "id":1, "method":"oauth2.revokeUserApplication",
"params":{"service_key":"m9f5kyzfjjrssz6d5sfkbuw3", "client": {
"client_id":"yq7chp6ufkeea587gp6wqa6x", "client_secret":"Azn4Xa" },
"user_context":"testUser1455730448397" }, "jsonrpc":"2.0"}' -H 'Host:
api.mashery.com' -H 'Content-Type: application/json'
'http://X.X.99.100:80/oauth/authorization/?api_
key=8qavf83pg2uzgndtxz2w4dg3'
* Trying X.X.99.100...
* Connected to X.X.99.100 (X.X.99.100) port 80 (#0)
> POST /oauth/auth/v2/?api_key=8qavf83pg2uzgndtxz2w4dg3 HTTP/1.1
> Host: api.mashery.com
> User-Agent: curl/7.43.0
> Accept: */*
> Content-Type: application/json
> Content-Length: 241
>
* upload completely sent off: 241 out of 241 bytes
< HTTP/1.1 500 Server Error
< X-Mashery-Responder: 8abfe284898c
< Content-Type: application/json; charset=UTF-8
< Cache-Control: no-store
< Content-Length: 81
< Date: Tue, 30 Oct 2018 16:41:56 GMT
<
* Connection #0 to host X.X.99.100 left intact

```

```
{"jsonrpc":"2.0","id":1,"error":{"message":"Internal Server Error","code":-2001}}
```

## Refresh token

```
curl -k -v -d '{"id":1,"method":"oauth2.createAccessToken","params":
{"service_key":"m9f5kyzfjjrssz6d5sfkbuw3","client":{"client_
id":"yq7chp6ufkeea587gp6wqa6x","client_secret":"Azn4Xa"},"token_data":
{"grant_type":"refresh_token","refresh_
token":"8f9qy6jrw3r23j2r6wqwp2jh"},"uri":{"redirect_
uri":"https://some.com/cb"},"user_context":null},"jsonrpc":"2.0"}' -H
'Host: api.mashery.com' -H 'Content-Type: application/json'
'http://X.X.99.100:80/oauth/authorization/?api_
key=8qavf83pg2uzgndtxz2w4dg3'
* Trying X.X.99.100...
* Connected to X.X.99.100 (X.X.99.100) port 80 (#0)
> POST /oauth/auth/v2/?api_key=8qavf83pg2uzgndtxz2w4dg3 HTTP/1.1
> Host: api.mashery.com
> User-Agent: curl/7.43.0
> Accept: */*
> Content-Type: application/json
> Content-Length: 339
>
* upload completely sent off: 339 out of 339 bytes
< HTTP/1.1 200 OK
< X-Mashery-Responder: 8abfe284898c
< Content-Type: application/json; charset=UTF-8
< Cache-Control: no-store
< Content-Length: 274
< Date: Tue, 30 Oct 2018 16:55:36 GMT
<
* Connection #0 to host X.X.99.100 left intact
{"jsonrpc":"2.0","id":1,"result":{"token_type":"bearer","return_
type":"json","access_token":"79rkkkxd4rk2zzbrqj4z93vb","expires_
in":36000,"refresh_token":"7db67za3xvdg99nc7gnctkqw","scope":null,"user_
context":"testUser1455730448397","uri":null,"extended":null,"state":nul
l}}
```

## Sample direct calls with Basic Authentication

### Create access token

```
curl -v -d '{"id":1,"method":"oauth2.createAccessToken","params":
{"service_key":"m9f5kyzfjjrssz6d5sfkbuw3","client":{"client_
id":"yq7chp6ufkeea587gp6wqa6x","client_secret":"Azn4Xa"},"token_data":
```

```

{"grant_type":"implicit"},"uri":{"redirect_
uri":"https://some.com/cb"},"user_
context":"testUser1455730448397"},"jsonrpc":"2.0"}'
'https://X.X.57.97:8083/v2/json-rpc' -u root:changeme -k
* Trying X.X.57.97...
* Connected to X.X.57.97 (X.X.57.97) port 8083 (#0)
* found 148 certificates in /etc/ssl/certs/ca-certificates.crt
* found 592 certificates in /etc/ssl/certs
* ALPN, offering http/1.1
* SSL connection using TLS1.2 / ECDHE_RSA_AES_128_GCM_SHA256
*     server certificate verification SKIPPED
*     server certificate status verification SKIPPED
*     common name: api.example.com (does not match 'X.X.57.97')
*     server certificate expiration date OK
*     server certificate activation date OK
*     certificate public key: RSA
*     certificate version: #3
*     subject:
C=defC,ST=defST,L=defL,O=defO,OU=defOU,CN=api.example.com
*     start date: Wed, 14 Nov 2018 03:28:23 GMT
*     expire date: Fri, 21 Oct 2118 03:28:23 GMT
*     issuer:
C=defC,ST=defST,L=defL,O=defO,OU=defOU,CN=api.example.com
*     compression: NULL
* ALPN, server did not agree to a protocol
* Server auth using Basic with user 'root'
> POST /v2/json-rpc HTTP/1.1
> Host: X.X.57.97:8083
> Authorization: Basic cm9vdDpjaGFuZ2VtZQ==
> User-Agent: curl/7.47.0
> Accept: */*
> Content-Length: 310
> Content-Type: application/x-www-form-urlencoded
>
* upload completely sent off: 310 out of 310 bytes
< HTTP/1.1 200 OK
< Content-Type: application/json;charset=UTF-8
< Cache-Control: no-store
< Content-Length: 231
< Server: Jetty(8.1.3.v20120522)
<
* Connection #0 to host X.X.57.97 left intact
{"jsonrpc":"2.0","id":1,"result":{"token_type":"bearer","return_
type":"json","access_token":"q3nxsetjry582yhq2ej3xa7j","expires_
in":36000,"scope":null,"user_

```

```
context":"testUser1455730448397","uri":null,"extended":null,"state":null}
```

### Create authorization code

```
curl -v -d '{ "id":1,"method":"oauth2.createAuthorizationCode","params":
{"service_key":"m9f5kyzfjjrssz6d5sfkbuw3","client":{"client_
id":"yq7chp6ufkeea587gp6wqa6x"},"response_type":"code","uri":{"redirect_
uri":"https://some.com/cb"},"user_context":
"testUser1455730448397"},"jsonrpc":"2.0"}'
'https://X.X.57.97:8083/v2/json-rpc' -u root:changeme -k
* Trying X.X.57.97...
* Connected to X.X.57.97 (X.X.57.97) port 8083 (#0)
* found 148 certificates in /etc/ssl/certs/ca-certificates.crt
* found 592 certificates in /etc/ssl/certs
* ALPN, offering http/1.1
* SSL connection using TLS1.2 / ECDHE_RSA_AES_128_GCM_SHA256
*    server certificate verification SKIPPED
*    server certificate status verification SKIPPED
*    common name: api.example.com (does not match 'X.X.57.97')
*    server certificate expiration date OK
*    server certificate activation date OK
*    certificate public key: RSA
*    certificate version: #3
*    subject:
C=defC,ST=defST,L=defL,O=defO,OU=defOU,CN=api.example.com
*    start date: Wed, 14 Nov 2018 03:28:23 GMT
*    expire date: Fri, 21 Oct 2118 03:28:23 GMT
*    issuer:
C=defC,ST=defST,L=defL,O=defO,OU=defOU,CN=api.example.com
*    compression: NULL
* ALPN, server did not agree to a protocol
* Server auth using Basic with user 'root'
> POST /v2/json-rpc HTTP/1.1
> Host: X.X.57.97:8083
> Authorization: Basic cm9vdDpjaGFuZ2VtZQ==
> User-Agent: curl/7.47.0
> Accept: */*
> Content-Length: 277
> Content-Type: application/x-www-form-urlencoded
>
* upload completely sent off: 277 out of 277 bytes
< HTTP/1.1 200 OK
< Content-Type: application/json;charset=UTF-8
< Cache-Control: no-store
< Content-Length: 168
```

```
< Server: Jetty(8.1.3.v20120522)
<
* Connection #0 to host X.X.57.97 left intact
{"jsonrpc":"2.0","id":1,"result":
{"code":"33benr9fb9vah3g8ys7scmh5","uri":{"redirect_
uri":"https://some.com/cb?code=33benr9fb9vah3g8ys7scmh5","state":""}},"e
rror":null}
```

## Create access token with authorization code

```
curl -v -d '{"id":1,"method":"oauth2.createAccessToken","params":
{"service_key":"m9f5kyzfjjrssh6d5sfkbuw3","client":{"client_
id":"yq7chp6ufkeea587gp6wqa6x","client_secret":"Azn4Xa"},"token_data":
{"grant_type":"authorization_
code","code":"33benr9fb9vah3g8ys7scmh5","scope":"scope1"},"uri":
{"redirect_uri":"https://some.com/cb"},"user_context":
"testUser1455730448397"},"jsonrpc":"2.0"}'
'https://X.X.57.97:8083/v2/json-rpc' -u root:changeme -k
* Trying X.X.57.97...
* Connected to X.X.57.97 (X.X.57.97) port 8083 (#0)
* found 148 certificates in /etc/ssl/certs/ca-certificates.crt
* found 592 certificates in /etc/ssl/certs
* ALPN, offering http/1.1
* SSL connection using TLS1.2 / ECDHE_RSA_AES_128_GCM_SHA256
* server certificate verification SKIPPED
* server certificate status verification SKIPPED
* common name: api.example.com (does not match 'X.X.57.97')
* server certificate expiration date OK
* server certificate activation date OK
* certificate public key: RSA
* certificate version: #3
* subject:
C=defC,ST=defST,L=defL,O=defO,OU=defOU,CN=api.example.com
* start date: Wed, 14 Nov 2018 03:28:23 GMT
* expire date: Fri, 21 Oct 2118 03:28:23 GMT
* issuer:
C=defC,ST=defST,L=defL,O=defO,OU=defOU,CN=api.example.com
* compression: NULL
* ALPN, server did not agree to a protocol
* Server auth using Basic with user 'root'
> POST /v2/json-rpc HTTP/1.1
> Host: X.X.57.97:8083
> Authorization: Basic cm9vdDpjaGFuZ2VtZQ==
> User-Agent: curl/7.47.0
> Accept: */*
> Content-Length: 372
```



```
> Content-Type: application/x-www-form-urlencoded
>
* upload completely sent off: 372 out of 372 bytes
< HTTP/1.1 200 OK
< Content-Type: application/json;charset=UTF-8
< Cache-Control: no-store
< Content-Length: 274
< Server: Jetty(8.1.3.v20120522)
<
* Connection #0 to host X.X.57.97 left intact
{"jsonrpc":"2.0","id":1,"result":{"token_type":"bearer","return_
type":"json","access_token":"s4b7qs24qfns2m3ecjmuvgh4","expires_
in":36000,"refresh_token":"xekntywdyp2hg7yvfxbac5g","scope":null,"user_
context":"testUser1455730448397","uri":null,"extended":null,"state":nul
l}}
```

## Update access token

```
curl -v -d '{"id":1,"method":"oauth2.updateAccessToken","params":
{"service_key":"m9f5kyzfjjrssz6d5sfkbw3","client":{"client_
id":"yq7chp6ufkeea587gp6wqa6x","client_secret":"Azn4Xa"},"access_
token":"s4b7qs24qfns2m3ecjmuvgh4","user_
context":"testUser1455730448397", "expires_in":600},"jsonrpc":"2.0"}'
'https://X.X.57.97:8083/v2/json-rpc' -u root:changeme -k
* Trying X.X.57.97...
* Connected to X.X.57.97 (X.X.57.97) port 8083 (#0)
* found 148 certificates in /etc/ssl/certs/ca-certificates.crt
* found 592 certificates in /etc/ssl/certs
* ALPN, offering http/1.1
* SSL connection using TLS1.2 / ECDHE_RSA_AES_128_GCM_SHA256
* server certificate verification SKIPPED
* server certificate status verification SKIPPED
* common name: api.example.com (does not match 'X.X.57.97')
* server certificate expiration date OK
* server certificate activation date OK
* certificate public key: RSA
* certificate version: #3
* subject:
C=defC,ST=defST,L=defL,O=defO,OU=defOU,CN=api.example.com
* start date: Wed, 14 Nov 2018 03:28:23 GMT
* expire date: Fri, 21 Oct 2118 03:28:23 GMT
* issuer:
C=defC,ST=defST,L=defL,O=defO,OU=defOU,CN=api.example.com
* compression: NULL
* ALPN, server did not agree to a protocol
* Server auth using Basic with user 'root'
```

```

> POST /v2/json-rpc HTTP/1.1
> Host: X.X.57.97:8083
> Authorization: Basic cm9vdDpjaGFuZ2VtZQ==
> User-Agent: curl/7.47.0
> Accept: */*
> Content-Length: 287
> Content-Type: application/x-www-form-urlencoded
>
* upload completely sent off: 287 out of 287 bytes
< HTTP/1.1 200 OK
< Content-Type: application/json;charset=UTF-8
< Cache-Control: no-store
< Content-Length: 322
< Server: Jetty(8.1.3.v20120522)
<
* Connection #0 to host X.X.57.97 left intact
{"jsonrpc":"2.0","id":1,"result":{"token_type":"bearer","access_token":"s4b7qs24qfns2m3ecjmuvg4","expires":"2018-11-21T10:57:43Z","refresh_token_expires":1542765336403,"scope":null,"user_context":"testUser1455730448397","uri":null,"grant_type":"authorization_code","client_id":"yq7chp6ufkeea587gp6wqa6x","extended":null}}

```

## Fetch access token

```

curl -v -d '{"id": 1,"jsonrpc":"2.0","method":
"oauth2.fetchAccessToken","params": {"service_
key":"m9f5kyzfjrrsz6d5sfkbuw3","access_token":
"s4b7qs24qfns2m3ecjmuvg4"}}' 'https://X.X.57.97:8083/v2/json-rpc' -u
root:changeme -k
* Trying X.X.57.97...
* Connected to X.X.57.97 (X.X.57.97) port 8083 (#0)
* found 148 certificates in /etc/ssl/certs/ca-certificates.crt
* found 592 certificates in /etc/ssl/certs
* ALPN, offering http/1.1
* SSL connection using TLS1.2 / ECDHE_RSA_AES_128_GCM_SHA256
* server certificate verification SKIPPED
* server certificate status verification SKIPPED
* common name: api.example.com (does not match 'X.X.57.97')
* server certificate expiration date OK
* server certificate activation date OK
* certificate public key: RSA
* certificate version: #3
* subject:
C=defC,ST=defST,L=defL,O=defO,OU=defOU,CN=api.example.com
* start date: Wed, 14 Nov 2018 03:28:23 GMT
* expire date: Fri, 21 Oct 2118 03:28:23 GMT

```

```

*       issuer:
C=defC,ST=defST,L=defL,O=defO,OU=defOU,CN=api.example.com
*       compression: NULL
* ALPN, server did not agree to a protocol
* Server auth using Basic with user 'root'
> POST /v2/json-rpc HTTP/1.1
> Host: X.X.57.97:8083
> Authorization: Basic cm9vdDpjaGFuZ2VtZQ==
> User-Agent: curl/7.47.0
> Accept: */*
> Content-Length: 157
> Content-Type: application/x-www-form-urlencoded
>
* upload completely sent off: 157 out of 157 bytes
< HTTP/1.1 200 OK
< Content-Type: application/json;charset=UTF-8
< Cache-Control: no-store
< Content-Length: 242
< Server: Jetty(8.1.3.v20120522)
<
* Connection #0 to host X.X.57.97 left intact
{"jsonrpc":"2.0","id":1,"result":{"token_type":"bearer","expires":"2018-11-21T10:57:43Z","scope":null,"user_context":"testUser1455730448397","uri":null,"grant_type":"authorization_code","client_id":"yq7chp6ufkeea587gp6wqa6x","extended":null}}

```

## Fetch application

```

curl -v -d '{"id":
1,"jsonrpc":"2.0","method":"oauth2.fetchApplication","params":{"service_
key":"m9f5kyzfjrrsz6d5sfkbuw3","client":{"client_
id":"yq7chp6ufkeea587gp6wqa6x","client_secret":"Azn4Xa"}}}'
'https://X.X.57.97:8083/v2/json-rpc' -u root:changeme -k
*   Trying X.X.57.97...
* Connected to X.X.57.97 (X.X.57.97) port 8083 (#0)
* found 148 certificates in /etc/ssl/certs/ca-certificates.crt
* found 592 certificates in /etc/ssl/certs
* ALPN, offering http/1.1
* SSL connection using TLS1.2 / ECDHE_RSA_AES_128_GCM_SHA256
*       server certificate verification SKIPPED
*       server certificate status verification SKIPPED
*       common name: api.example.com (does not match 'X.X.57.97')
*       server certificate expiration date OK
*       server certificate activation date OK
*       certificate public key: RSA
*       certificate version: #3

```

```

*      subject:
C=defC,ST=defST,L=defL,O=defO,OU=defOU,CN=api.example.com
*      start date: Wed, 14 Nov 2018 03:28:23 GMT
*      expire date: Fri, 21 Oct 2118 03:28:23 GMT
*      issuer:
C=defC,ST=defST,L=defL,O=defO,OU=defOU,CN=api.example.com
*      compression: NULL
* ALPN, server did not agree to a protocol
* Server auth using Basic with user 'root'
> POST /v2/json-rpc HTTP/1.1
> Host: X.X.57.97:8083
> Authorization: Basic cm9vdDpjaGFuZ2VtZQ==
> User-Agent: curl/7.47.0
> Accept: */*
> Content-Length: 187
> Content-Type: application/x-www-form-urlencoded
>
* upload completely sent off: 187 out of 187 bytes
< HTTP/1.1 200 OK
< Content-Type: application/json;charset=UTF-8
< Cache-Control: no-store
< Content-Length: 82
< Server: Jetty(8.1.3.v20120522)
<
* Connection #0 to host X.X.57.97 left intact
{"jsonrpc":"2.0","id":1,"result":
{"id":251547,"name":"TestOAuthRegisterCallBack"}}

```

## Fetch user application

```

curl -v -d '{ "id":1, "method":"oauth2.fetchUserApplications",
"params":{"service_key":"m9f5kyzfjjrssz6d5sfkbw3", "user_
context":"testUser1455730448397" }, "jsonrpc":"2.0"}'
'https://X.X.57.97:8083/v2/json-rpc' -u root:changeme -k
* Trying X.X.57.97...
* Connected to X.X.57.97 (X.X.57.97) port 8083 (#0)
* found 148 certificates in /etc/ssl/certs/ca-certificates.crt
* found 592 certificates in /etc/ssl/certs
* ALPN, offering http/1.1
* SSL connection using TLS1.2 / ECDHE_RSA_AES_128_GCM_SHA256
*      server certificate verification SKIPPED
*      server certificate status verification SKIPPED
*      common name: api.example.com (does not match 'X.X.57.97')
*      server certificate expiration date OK
*      server certificate activation date OK
*      certificate public key: RSA

```

```

*      certificate version: #3
*      subject:
C=defC,ST=defST,L=defL,O=defO,OU=defOU,CN=api.example.com
*      start date: Wed, 14 Nov 2018 03:28:23 GMT
*      expire date: Fri, 21 Oct 2118 03:28:23 GMT
*      issuer:
C=defC,ST=defST,L=defL,O=defO,OU=defOU,CN=api.example.com
*      compression: NULL
* ALPN, server did not agree to a protocol
* Server auth using Basic with user 'root'
> POST /v2/json-rpc HTTP/1.1
> Host: X.X.57.97:8083
> Authorization: Basic cm9vdDpjaGFuZ2VtZQ==
> User-Agent: curl/7.47.0
> Accept: */*
> Content-Length: 162
> Content-Type: application/x-www-form-urlencoded
>
* upload completely sent off: 162 out of 162 bytes
< HTTP/1.1 200 OK
< Content-Type: application/json;charset=UTF-8
< Cache-Control: no-store
< Content-Length: 195
< Server: Jetty(8.1.3.v20120522)
<
* Connection #0 to host X.X.57.97 left intact
{"jsonrpc":"2.0","id":1,"result":
[{"id":251547,"name":"TestOAuthRegisterCallBack","access_tokens":
["s4b7qs24qfns2m3ecjmuvgh4","q3nxsetjry582yhq2ej3xa7j"],"client_
id":"yq7chp6ufkeea587gp6wqa6x"]}]}

```

## Revoke access token

```

curl -v -d '{ "id":1, "method":"oauth2.revokeAccessToken", "params":{
"service_key":"m9f5kyzfjrrsz6d5sfkbuw3", "client": { "client_
id":"yq7chp6ufkeea587gp6wqa6x", "client_secret":"Azn4Xa" }, "access_
token":"s4b7qs24qfns2m3ecjmuvgh4" }, "jsonrpc":"2.0"}'
'https://X.X.57.97:8083/v2/json-rpc' -u root:changeme -k
* Trying X.X.57.97...
* Connected to X.X.57.97 (X.X.57.97) port 8083 (#0)
* found 148 certificates in /etc/ssl/certs/ca-certificates.crt
* found 592 certificates in /etc/ssl/certs
* ALPN, offering http/1.1
* SSL connection using TLS1.2 / ECDHE_RSA_AES_128_GCM_SHA256
*      server certificate verification SKIPPED
*      server certificate status verification SKIPPED

```

```

*      common name: api.example.com (does not match 'X.X.57.97')
*      server certificate expiration date OK
*      server certificate activation date OK
*      certificate public key: RSA
*      certificate version: #3
*      subject:
C=defC,ST=defST,L=defL,O=defO,OU=defOU,CN=api.example.com
*      start date: Wed, 14 Nov 2018 03:28:23 GMT
*      expire date: Fri, 21 Oct 2118 03:28:23 GMT
*      issuer:
C=defC,ST=defST,L=defL,O=defO,OU=defOU,CN=api.example.com
*      compression: NULL
* ALPN, server did not agree to a protocol
* Server auth using Basic with user 'root'
> POST /v2/json-rpc HTTP/1.1
> Host: X.X.57.97:8083
> Authorization: Basic cm9vdDpjaGFuZ2VtZQ==
> User-Agent: curl/7.47.0
> Accept: */*
> Content-Length: 240
> Content-Type: application/x-www-form-urlencoded
>
* upload completely sent off: 240 out of 240 bytes
< HTTP/1.1 200 OK
< Content-Type: application/json; charset=UTF-8
< Cache-Control: no-store
< Content-Length: 38
< Server: Jetty(8.1.3.v20120522)
<
* Connection #0 to host X.X.57.97 left intact
{"jsonrpc":"2.0","id":1,"result":true}

```

## Revoke user application

```

curl -v -d '{ "id":1, "method":"oauth2.revokeUserApplication",
"params":{"service_key":"m9f5kyzfjjrssz6d5sfkbw3", "client": {
"client_id":"yq7chp6ufkeea587gp6wqa6x", "client_secret":"Azn4Xa" },
"user_context":"testUser1455730448397" }, "jsonrpc":"2.0"}'
'https://X.X.57.97:8083/v2/json-rpc' -u root:changeme -k
* Trying X.X.57.97...
* Connected to X.X.57.97 (X.X.57.97) port 8083 (#0)
* found 148 certificates in /etc/ssl/certs/ca-certificates.crt
* found 592 certificates in /etc/ssl/certs
* ALPN, offering http/1.1
* SSL connection using TLS1.2 / ECDHE_RSA_AES_128_GCM_SHA256
*      server certificate verification SKIPPED

```

```

*      server certificate status verification SKIPPED
*      common name: api.example.com (does not match 'X.X.57.97')
*      server certificate expiration date OK
*      server certificate activation date OK
*      certificate public key: RSA
*      certificate version: #3
*      subject:
C=defC,ST=defST,L=defL,O=defO,OU=defOU,CN=api.example.com
*      start date: Wed, 14 Nov 2018 03:28:23 GMT
*      expire date: Fri, 21 Oct 2118 03:28:23 GMT
*      issuer:
C=defC,ST=defST,L=defL,O=defO,OU=defOU,CN=api.example.com
*      compression: NULL
* ALPN, server did not agree to a protocol
* Server auth using Basic with user 'root'
> POST /v2/json-rpc HTTP/1.1
> Host: X.X.57.97:8083
> Authorization: Basic cm9vdDpjaGFuZ2VtZQ==
> User-Agent: curl/7.47.0
> Accept: */*
> Content-Length: 241
> Content-Type: application/x-www-form-urlencoded
>
* upload completely sent off: 241 out of 241 bytes
< HTTP/1.1 200 OK
< Content-Type: application/json; charset=UTF-8
< Cache-Control: no-store
< Content-Length: 38
< Server: Jetty(8.1.3.v20120522)
<
* Connection #0 to host X.X.57.97 left intact
{"jsonrpc":"2.0","id":1,"result":true}

```

## Refresh token

```

curl -k -v -d '{"id":1,"method":"oauth2.createAccessToken","params":
{"service_key":"m9f5kyzfjjrssz6d5sfkbw3","client":{"client_
id":"yq7chp6ufkeea587gp6wqa6x","client_secret":"Azn4Xa"},"token_data":
{"grant_type":"refresh_token","refresh_
token":"xekntywdyp2hg7yvfboxac5g"},"uri":{"redirect_
uri":"https://some.com/cb"},"user_context":null},"jsonrpc":"2.0"}'
'https://X.X.57.97:8083/v2/json-rpc' -u root:changeme -k
* Trying X.X.57.97...
* Connected to X.X.57.97 (X.X.57.97) port 8083 (#0)
* found 148 certificates in /etc/ssl/certs/ca-certificates.crt
* found 592 certificates in /etc/ssl/certs

```

```

* ALPN, offering http/1.1
* SSL connection using TLS1.2 / ECDHE_RSA_AES_128_GCM_SHA256
*       server certificate verification SKIPPED
*       server certificate status verification SKIPPED
*       common name: api.example.com (does not match 'X.X.57.97')
*       server certificate expiration date OK
*       server certificate activation date OK
*       certificate public key: RSA
*       certificate version: #3
*       subject:
C=defC,ST=defST,L=defL,O=defO,OU=defOU,CN=api.example.com
*       start date: Wed, 14 Nov 2018 03:28:23 GMT
*       expire date: Fri, 21 Oct 2118 03:28:23 GMT
*       issuer:
C=defC,ST=defST,L=defL,O=defO,OU=defOU,CN=api.example.com
*       compression: NULL
* ALPN, server did not agree to a protocol
* Server auth using Basic with user 'root'
> POST /v2/json-rpc HTTP/1.1
> Host: X.X.57.97:8083
> Authorization: Basic cm9vdDpjaGFuZ2VtZQ==
> User-Agent: curl/7.47.0
> Accept: */*
> Content-Length: 339
> Content-Type: application/x-www-form-urlencoded
>
* upload completely sent off: 339 out of 339 bytes
< HTTP/1.1 200 OK
< Content-Type: application/json;charset=UTF-8
< Cache-Control: no-store
< Content-Length: 274
< Server: Jetty(8.1.3.v20120522)
<
* Connection #0 to host X.X.57.97 left intact
{"jsonrpc":"2.0","id":1,"result":{"token_type":"bearer","return_
type":"json","access_token":"z242k8wyshjmvjrkckask2f","expires_
in":36000,"refresh_token":"epecdr7zzfjxwwfu35395dx9","scope":null,"user_
context":"testUser1455730448397","uri":null,"extended":null,"state":nul
l}}

```



# Making OAuth 2.0 Calls

## OAuth API Supported Methods

The following methods support the OAuth 2.0 functionality:

- [fetchApplication](#) - Verifies that client has a valid identifier and fetches the application data (e.g., name) for a client requesting access to user resource.
- [createAuthorizationCode](#) - Creates authorization code that can be subsequently used to obtain an access token. Used in Authorization Code flow.
- [createAccessToken](#) - Creates access token for the flow indicated by response and grant type. Individual parameters are validated based on the indicated flow.
- [fetchAccessToken](#) - Fetches the specified access token and data associated with the token or returns that it is invalid.
- [fetchUserApplications](#) - Fetches the applications with access tokens for the given user.
- [revokeAccessToken](#) - Revokes the access token.
- [revokeUserApplication](#) - Revokes all tokens for the application for the specified user.
- [updateAccessToken](#) - Updates access token based on individual parameters specified.

## Sample Calls

### Sample Direct Calls using HTTPS and Basic Authentication

#### Create access token

```
curl -v -d '{"id":1,"method":"oauth2.createAccessToken","params":
{"service_key":"m9f5kyzfjjrssh6d5sfkbw3","client":{"client_
id":"yq7chp6ufkeea587gp6wqa6x","client_secret":"Azn4Xa"},"token_data":
{"grant_type":"implicit"},"uri":{"redirect_
uri":"https://some.com/cb"},"user_
context":"testUser1455730448397"},"jsonrpc":"2.0"}'
'https://x.x.57.97:8083/v3/json-rpc' -u root:changeme -k
* Trying x.x.57.97...
* Connected to x.x.57.97 (x.x.57.97) port 8083 (#0)
```

```

* found 148 certificates in /etc/ssl/certs/ca-certificates.crt
* found 592 certificates in /etc/ssl/certs
* ALPN, offering http/1.1
* SSL connection using TLS1.2 / ECDHE_RSA_AES_128_GCM_SHA256
*     server certificate verification SKIPPED
*     server certificate status verification SKIPPED
*     common name: api.example.com (does not match 'x.x.57.97')
*     server certificate expiration date OK
*     server certificate activation date OK
*     certificate public key: RSA
*     certificate version: #3
*     subject:
C=defC,ST=defST,L=defL,O=defO,OU=defOU,CN=api.example.com
*     start date: Wed, 14 Nov 2018 03:28:23 GMT
*     expire date: Fri, 21 Oct 2118 03:28:23 GMT
*     issuer:
C=defC,ST=defST,L=defL,O=defO,OU=defOU,CN=api.example.com
*     compression: NULL
* ALPN, server did not agree to a protocol
* Server auth using Basic with user 'root'
> POST /v3/json-rpc HTTP/1.1
> Host: x.x.57.97:8083
> Authorization: Basic cm9vdDpjaGFuZ2VtZQ==
> User-Agent: curl/7.47.0
> Accept: */*
> Content-Length: 310
> Content-Type: application/x-www-form-urlencoded
>
* upload completely sent off: 310 out of 310 bytes
< HTTP/1.1 200 OK
< Content-Type: application/json; charset=UTF-8
< Cache-Control: no-store
< Content-Length: 231
< Server: Jetty(8.1.3.v20120522)
<
* Connection #0 to host x.x.57.97 left intact
{"jsonrpc":"2.0","id":1,"result":{"token_type":"bearer","return_
type":"json","access_token":"q3nxsetjry582yhq2ej3xa7j","expires_
in":36000,"scope":null,"user_
context":"testUser1455730448397","uri":null,"extended":null,"state":nul
l}}

```

## Create authorization code

```

curl -v -d '{ "id":1,"method":"oauth2.createAuthorizationCode","params":
{"service_key":"m9f5kyzfjjrssz6d5sfkbuw3","client":{"client_

```

```

id":"yq7chp6ufkeea587gp6wqa6x"},"response_type":"code","uri":{"redirect_
uri":"https://some.com/cb"},"user_context":
"testUser1455730448397"},"jsonrpc":"2.0"}'
'https://x.x.57.97:8083/v3/json-rpc' -u root:changeme -k
* Trying x.x.57.97...
* Connected to x.x.57.97 (x.x.57.97) port 8083 (#0)
* found 148 certificates in /etc/ssl/certs/ca-certificates.crt
* found 592 certificates in /etc/ssl/certs
* ALPN, offering http/1.1
* SSL connection using TLS1.2 / ECDHE_RSA_AES_128_GCM_SHA256
*     server certificate verification SKIPPED
*     server certificate status verification SKIPPED
*     common name: api.example.com (does not match 'x.x.57.97')
*     server certificate expiration date OK
*     server certificate activation date OK
*     certificate public key: RSA
*     certificate version: #3
*     subject:
C=defC,ST=defST,L=defL,O=defO,OU=defOU,CN=api.example.com
*     start date: Wed, 14 Nov 2018 03:28:23 GMT
*     expire date: Fri, 21 Oct 2118 03:28:23 GMT
*     issuer:
C=defC,ST=defST,L=defL,O=defO,OU=defOU,CN=api.example.com
*     compression: NULL
* ALPN, server did not agree to a protocol
* Server auth using Basic with user 'root'
> POST /v3/json-rpc HTTP/1.1
> Host: x.x.57.97:8083
> Authorization: Basic cm9vdDpjaGFuZ2VtZQ==
> User-Agent: curl/7.47.0
> Accept: */*
> Content-Length: 277
> Content-Type: application/x-www-form-urlencoded
>
* upload completely sent off: 277 out of 277 bytes
< HTTP/1.1 200 OK
< Content-Type: application/json;charset=UTF-8
< Cache-Control: no-store
< Content-Length: 168
< Server: Jetty(8.1.3.v20120522)
<
* Connection #0 to host x.x.57.97 left intact
{"jsonrpc":"2.0","id":1,"result":
{"code":"33benr9fb9vah3g8ys7scmh5","uri":{"redirect_

```

```
uri":"https://some.com/cb?code=33benr9fb9vah3g8ys7scmh5","state":""}},"error":null}
```

## Create access token with authorization code

```
curl -v -d '{"id":1,"method":"oauth2.createAccessToken","params":
{"service_key":"m9f5kyzfjjrssz6d5sfkbuw3","client":{"client_
id":"yq7chp6ufkeea587gp6wqa6x","client_secret":"Azn4Xa"},"token_data":
{"grant_type":"authorization_
code","code":"33benr9fb9vah3g8ys7scmh5","scope":"scope1"},"uri":
{"redirect_uri":"https://some.com/cb"},"user_context":
{"testUser1455730448397"},"jsonrpc":"2.0"}'
'https://x.x.57.97:8083/v3/json-rpc' -u root:changeme -k
* Trying x.x.57.97...
* Connected to x.x.57.97 (x.x.57.97) port 8083 (#0)
* found 148 certificates in /etc/ssl/certs/ca-certificates.crt
* found 592 certificates in /etc/ssl/certs
* ALPN, offering http/1.1
* SSL connection using TLS1.2 / ECDHE_RSA_AES_128_GCM_SHA256
* server certificate verification SKIPPED
* server certificate status verification SKIPPED
* common name: api.example.com (does not match 'x.x.57.97')
* server certificate expiration date OK
* server certificate activation date OK
* certificate public key: RSA
* certificate version: #3
* subject:
C=defC,ST=defST,L=defL,O=defO,OU=defOU,CN=api.example.com
* start date: Wed, 14 Nov 2018 03:28:23 GMT
* expire date: Fri, 21 Oct 2118 03:28:23 GMT
* issuer:
C=defC,ST=defST,L=defL,O=defO,OU=defOU,CN=api.example.com
* compression: NULL
* ALPN, server did not agree to a protocol
* Server auth using Basic with user 'root'
> POST /v3/json-rpc HTTP/1.1
> Host: x.x.57.97:8083
> Authorization: Basic cm9vdDpjaGFuZ2VtZQ==
> User-Agent: curl/7.47.0
> Accept: */*
> Content-Length: 372
> Content-Type: application/x-www-form-urlencoded
>
* upload completely sent off: 372 out of 372 bytes
< HTTP/1.1 200 OK
< Content-Type: application/json; charset=UTF-8
```

```
< Cache-Control: no-store
< Content-Length: 274
< Server: Jetty(8.1.3.v20120522)
<
* Connection #0 to host x.x.57.97 left intact
{"jsonrpc":"2.0","id":1,"result":{"token_type":"bearer","return_type":"json","access_token":"s4b7qs24qfns2m3ecjmuvgh4","expires_in":36000,"refresh_token":"xekntywdyp2hg7yvfboxac5g","scope":null,"user_context":"testUser1455730448397","uri":null,"extended":null,"state":null}}
```

## Update access token

```
curl -v -d '{"id":1,"method":"oauth2.updateAccessToken","params":{"service_key":"m9f5kyzfjjrssz6d5sfkbuw3","client":{"client_id":"yq7chp6ufkeea587gp6wqa6x","client_secret":"Azn4Xa"},"access_token":"s4b7qs24qfns2m3ecjmuvgh4","user_context":"testUser1455730448397","expires_in":600},"jsonrpc":"2.0"}'
'https://x.x.57.97:8083/v3/json-rpc' -u root:changeme -k
* Trying x.x.57.97...
* Connected to x.x.57.97 (x.x.57.97) port 8083 (#0)
* found 148 certificates in /etc/ssl/certs/ca-certificates.crt
* found 592 certificates in /etc/ssl/certs
* ALPN, offering http/1.1
* SSL connection using TLS1.2 / ECDHE_RSA_AES_128_GCM_SHA256
* server certificate verification SKIPPED
* server certificate status verification SKIPPED
* common name: api.example.com (does not match 'x.x.57.97')
* server certificate expiration date OK
* server certificate activation date OK
* certificate public key: RSA
* certificate version: #3
* subject:
C=defC,ST=defST,L=defL,O=defO,OU=defOU,CN=api.example.com
* start date: Wed, 14 Nov 2018 03:28:23 GMT
* expire date: Fri, 21 Oct 2118 03:28:23 GMT
* issuer:
C=defC,ST=defST,L=defL,O=defO,OU=defOU,CN=api.example.com
* compression: NULL
* ALPN, server did not agree to a protocol
* Server auth using Basic with user 'root'
> POST /v3/json-rpc HTTP/1.1
> Host: x.x.57.97:8083
> Authorization: Basic cm9vdDpjaGFuZ2VtZQ==
> User-Agent: curl/7.47.0
> Accept: */*
```

```
> Content-Length: 287
> Content-Type: application/x-www-form-urlencoded
>
* upload completely sent off: 287 out of 287 bytes
< HTTP/1.1 200 OK
< Content-Type: application/json;charset=UTF-8
< Cache-Control: no-store
< Content-Length: 322
< Server: Jetty(8.1.3.v20120522)
<
* Connection #0 to host x.x.57.97 left intact
{"jsonrpc":"2.0","id":1,"result":{"token_type":"bearer","access_token":"s4b7qs24qfns2m3ecjmuvg4","expires":"2018-11-21T10:57:43Z","refresh_token_expires":1542765336403,"scope":null,"user_context":"testUser1455730448397","uri":null,"grant_type":"authorization_code","client_id":"yq7chp6ufkeea587gp6wqa6x","extended":null}}
```

## Fetch access token

```
curl -v -d '{"id": 1,"jsonrpc":"2.0","method":
"oauth2.fetchAccessToken","params": {"service_
key":"m9f5kyzfjjrssz6d5sfkbuw3","access_token":
"s4b7qs24qfns2m3ecjmuvg4"}}' 'https://x.x.57.97:8083/v3/json-rpc' -u
root:changeme -k
* Trying x.x.57.97...
* Connected to x.x.57.97 (x.x.57.97) port 8083 (#0)
* found 148 certificates in /etc/ssl/certs/ca-certificates.crt
* found 592 certificates in /etc/ssl/certs
* ALPN, offering http/1.1
* SSL connection using TLS1.2 / ECDHE_RSA_AES_128_GCM_SHA256
* server certificate verification SKIPPED
* server certificate status verification SKIPPED
* common name: api.example.com (does not match 'x.x.57.97')
* server certificate expiration date OK
* server certificate activation date OK
* certificate public key: RSA
* certificate version: #3
* subject:
C=defC,ST=defST,L=defL,O=defO,OU=defOU,CN=api.example.com
* start date: Wed, 14 Nov 2018 03:28:23 GMT
* expire date: Fri, 21 Oct 2118 03:28:23 GMT
* issuer:
C=defC,ST=defST,L=defL,O=defO,OU=defOU,CN=api.example.com
* compression: NULL
* ALPN, server did not agree to a protocol
* Server auth using Basic with user 'root'
```

```

> POST /v3/json-rpc HTTP/1.1
> Host: x.x.57.97:8083
> Authorization: Basic cm9vdDpjaGFuZ2VtZQ==
> User-Agent: curl/7.47.0
> Accept: */*
> Content-Length: 157
> Content-Type: application/x-www-form-urlencoded
>
* upload completely sent off: 157 out of 157 bytes
< HTTP/1.1 200 OK
< Content-Type: application/json;charset=UTF-8
< Cache-Control: no-store
< Content-Length: 242
< Server: Jetty(8.1.3.v20120522)
<
* Connection #0 to host x.x.57.97 left intact
{"jsonrpc":"2.0","id":1,"result":{"token_type":"bearer","expires":"2018-11-21T10:57:43Z","scope":null,"user_context":"testUser1455730448397","uri":null,"grant_type":"authorization_code","client_id":"yq7chp6ufkeea587gp6wqa6x","extended":null}}

```

## Fetch application

```

curl -v -d '{"id":1,"jsonrpc":"2.0","method":"oauth2.fetchApplication","params":{"service_key":"m9f5kyzfjrrssz6d5sfkbuw3","client":{"client_id":"yq7chp6ufkeea587gp6wqa6x","client_secret":"Azn4Xa"}}}'
'https://x.x.57.97:8083/v3/json-rpc' -u root:changeme -k
* Trying x.x.57.97...
* Connected to x.x.57.97 (x.x.57.97) port 8083 (#0)
* found 148 certificates in /etc/ssl/certs/ca-certificates.crt
* found 592 certificates in /etc/ssl/certs
* ALPN, offering http/1.1
* SSL connection using TLS1.2 / ECDHE_RSA_AES_128_GCM_SHA256
*    server certificate verification SKIPPED
*    server certificate status verification SKIPPED
*    common name: api.example.com (does not match 'x.x.57.97')
*    server certificate expiration date OK
*    server certificate activation date OK
*    certificate public key: RSA
*    certificate version: #3
*    subject:
C=defC,ST=defST,L=defL,O=defO,OU=defOU,CN=api.example.com
*    start date: Wed, 14 Nov 2018 03:28:23 GMT
*    expire date: Fri, 21 Oct 2118 03:28:23 GMT
*    issuer:

```

```

C=defC,ST=defST,L=defL,O=defO,OU=defOU,CN=api.example.com
*       compression: NULL
* ALPN, server did not agree to a protocol
* Server auth using Basic with user 'root'
> POST /v3/json-rpc HTTP/1.1
> Host: x.x.57.97:8083
> Authorization: Basic cm9vdDpjaGFuZ2VtZQ==
> User-Agent: curl/7.47.0
> Accept: */*
> Content-Length: 187
> Content-Type: application/x-www-form-urlencoded
>
* upload completely sent off: 187 out of 187 bytes
< HTTP/1.1 200 OK
< Content-Type: application/json;charset=UTF-8
< Cache-Control: no-store
< Content-Length: 82
< Server: Jetty(8.1.3.v20120522)
<
* Connection #0 to host x.x.57.97 left intact
{"jsonrpc":"2.0","id":1,"result":
{"id":251547,"name":"TestOAuthRegisterCallBack"}}

```

## Fetch user application

```

curl -v -d '{ "id":1, "method":"oauth2.fetchUserApplications",
"params":{"service_key":"m9f5kyzfjjrssz6d5sfkbuw3", "user_
context":"testUser1455730448397" }, "jsonrpc":"2.0"}'
'https://x.x.57.97:8083/v3/json-rpc' -u root:changeme -k
*   Trying x.x.57.97...
* Connected to x.x.57.97 (x.x.57.97) port 8083 (#0)
* found 148 certificates in /etc/ssl/certs/ca-certificates.crt
* found 592 certificates in /etc/ssl/certs
* ALPN, offering http/1.1
* SSL connection using TLS1.2 / ECDHE_RSA_AES_128_GCM_SHA256
*       server certificate verification SKIPPED
*       server certificate status verification SKIPPED
*       common name: api.example.com (does not match 'x.x.57.97')
*       server certificate expiration date OK
*       server certificate activation date OK
*       certificate public key: RSA
*       certificate version: #3
*       subject:
C=defC,ST=defST,L=defL,O=defO,OU=defOU,CN=api.example.com
*       start date: Wed, 14 Nov 2018 03:28:23 GMT
*       expire date: Fri, 21 Oct 2118 03:28:23 GMT

```



```

*       issuer:
C=defC,ST=defST,L=defL,O=defO,OU=defOU,CN=api.example.com
*       compression: NULL
* ALPN, server did not agree to a protocol
* Server auth using Basic with user 'root'
> POST /v3/json-rpc HTTP/1.1
> Host: x.x.57.97:8083
> Authorization: Basic cm9vdDpjaGFuZ2VtZQ==
> User-Agent: curl/7.47.0
> Accept: */*
> Content-Length: 162
> Content-Type: application/x-www-form-urlencoded
>
* upload completely sent off: 162 out of 162 bytes
< HTTP/1.1 200 OK
< Content-Type: application/json;charset=UTF-8
< Cache-Control: no-store
< Content-Length: 195
< Server: Jetty(8.1.3.v20120522)
<
* Connection #0 to host x.x.57.97 left intact
{"jsonrpc":"2.0","id":1,"result":
[{"id":251547,"name":"Test0AuthRegisterCallBack","access_tokens":
["s4b7qs24qfns2m3ecjmuvg4","q3nxsetjry582yhq2ej3xa7j"],"client_
id":"yq7chp6ufkeea587gp6wqa6x"}]}

```

## Revoke access token

```

curl -v -d '{ "id":1, "method":"oauth2.revokeAccessToken", "params":{
"service_key":"m9f5kyzfjjrssh6d5sfkbw3", "client": { "client_
id":"yq7chp6ufkeea587gp6wqa6x", "client_secret":"Azn4Xa" }, "access_
token":"s4b7qs24qfns2m3ecjmuvg4" }, "jsonrpc":"2.0"}'
'https://x.x.57.97:8083/v3/json-rpc' -u root:changeme -k
* Trying x.x.57.97...
* Connected to x.x.57.97 (x.x.57.97) port 8083 (#0)
* found 148 certificates in /etc/ssl/certs/ca-certificates.crt
* found 592 certificates in /etc/ssl/certs
* ALPN, offering http/1.1
* SSL connection using TLS1.2 / ECDHE_RSA_AES_128_GCM_SHA256
*       server certificate verification SKIPPED
*       server certificate status verification SKIPPED
*       common name: api.example.com (does not match 'x.x.57.97')
*       server certificate expiration date OK
*       server certificate activation date OK
*       certificate public key: RSA
*       certificate version: #3

```

```

*      subject:
C=defC,ST=defST,L=defL,O=defO,OU=defOU,CN=api.example.com
*      start date: Wed, 14 Nov 2018 03:28:23 GMT
*      expire date: Fri, 21 Oct 2118 03:28:23 GMT
*      issuer:
C=defC,ST=defST,L=defL,O=defO,OU=defOU,CN=api.example.com
*      compression: NULL
* ALPN, server did not agree to a protocol
* Server auth using Basic with user 'root'
> POST /v3/json-rpc HTTP/1.1
> Host: x.x.57.97:8083
> Authorization: Basic cm9vdDpjaGFuZ2VtZQ==
> User-Agent: curl/7.47.0
> Accept: */*
> Content-Length: 240
> Content-Type: application/x-www-form-urlencoded
>
* upload completely sent off: 240 out of 240 bytes
< HTTP/1.1 200 OK
< Content-Type: application/json;charset=UTF-8
< Cache-Control: no-store
< Content-Length: 38
< Server: Jetty(8.1.3.v20120522)
<
* Connection #0 to host x.x.57.97 left intact
{"jsonrpc":"2.0","id":1,"result":true}

```

## Revoke user application

```

curl -v -d '{ "id":1, "method":"oauth2.revokeUserApplication",
"params":{"service_key":"m9f5kyzfjrrssz6d5sfkbuw3", "client": {
"client_id":"yq7chp6ufkeea587gp6wqa6x", "client_secret":"Azn4Xa" },
"user_context":"testUser1455730448397" }, "jsonrpc":"2.0"}'
'https://x.x.57.97:8083/v3/json-rpc' -u root:changeme -k
* Trying x.x.57.97...
* Connected to x.x.57.97 (x.x.57.97) port 8083 (#0)
* found 148 certificates in /etc/ssl/certs/ca-certificates.crt
* found 592 certificates in /etc/ssl/certs
* ALPN, offering http/1.1
* SSL connection using TLS1.2 / ECDHE_RSA_AES_128_GCM_SHA256
*      server certificate verification SKIPPED
*      server certificate status verification SKIPPED
*      common name: api.example.com (does not match 'x.x.57.97')
*      server certificate expiration date OK
*      server certificate activation date OK
*      certificate public key: RSA

```

```

*      certificate version: #3
*      subject:
C=defC,ST=defST,L=defL,O=defO,OU=defOU,CN=api.example.com
*      start date: Wed, 14 Nov 2018 03:28:23 GMT
*      expire date: Fri, 21 Oct 2118 03:28:23 GMT
*      issuer:
C=defC,ST=defST,L=defL,O=defO,OU=defOU,CN=api.example.com
*      compression: NULL
* ALPN, server did not agree to a protocol
* Server auth using Basic with user 'root'
> POST /v3/json-rpc HTTP/1.1
> Host: x.x.57.97:8083
> Authorization: Basic cm9vdDpjaGFuZ2VtZQ==
> User-Agent: curl/7.47.0
> Accept: */*
> Content-Length: 241
> Content-Type: application/x-www-form-urlencoded
>
* upload completely sent off: 241 out of 241 bytes
< HTTP/1.1 200 OK
< Content-Type: application/json;charset=UTF-8
< Cache-Control: no-store
< Content-Length: 38
< Server: Jetty(8.1.3.v20120522)
<
* Connection #0 to host x.x.57.97 left intact
{"jsonrpc":"2.0","id":1,"result":true}

```

## Refresh token

```

curl -k -v -d '{"id":1,"method":"oauth2.createAccessToken","params":
{"service_key":"m9f5kyzfjjrssz6d5sfkbuw3","client":{"client_
id":"yq7chp6ufkeea587gp6wqa6x","client_secret":"Azn4Xa"},"token_data":
{"grant_type":"refresh_token","refresh_
token":"xekntywdyp2hg7yvfbaxac5g"},"uri":{"redirect_
uri":"https://some.com/cb"},"user_context":null},"jsonrpc":"2.0"}'
'https://x.x.57.97:8083/v3/json-rpc' -u root:changeme -k
* Trying x.x.57.97...
* Connected to x.x.57.97 (x.x.57.97) port 8083 (#0)
* found 148 certificates in /etc/ssl/certs/ca-certificates.crt
* found 592 certificates in /etc/ssl/certs
* ALPN, offering http/1.1
* SSL connection using TLS1.2 / ECDHE_RSA_AES_128_GCM_SHA256
*      server certificate verification SKIPPED
*      server certificate status verification SKIPPED
*      common name: api.example.com (does not match 'x.x.57.97')

```

```

*      server certificate expiration date OK
*      server certificate activation date OK
*      certificate public key: RSA
*      certificate version: #3
*      subject:
C=defC,ST=defST,L=defL,O=defO,OU=defOU,CN=api.example.com
*      start date: Wed, 14 Nov 2018 03:28:23 GMT
*      expire date: Fri, 21 Oct 2118 03:28:23 GMT
*      issuer:
C=defC,ST=defST,L=defL,O=defO,OU=defOU,CN=api.example.com
*      compression: NULL
* ALPN, server did not agree to a protocol
* Server auth using Basic with user 'root'
> POST /v3/json-rpc HTTP/1.1
> Host: x.x.57.97:8083
> Authorization: Basic cm9vdDpjaGFuZ2VtZQ==
> User-Agent: curl/7.47.0
> Accept: */*
> Content-Length: 339
> Content-Type: application/x-www-form-urlencoded
>
* upload completely sent off: 339 out of 339 bytes
< HTTP/1.1 200 OK
< Content-Type: application/json;charset=UTF-8
< Cache-Control: no-store
< Content-Length: 274
< Server: Jetty(8.1.3.v20120522)
<
* Connection #0 to host x.x.57.97 left intact
{"jsonrpc":"2.0","id":1,"result":{"token_type":"bearer","return_
type":"json","access_token":"z242k8wyshjmvjrkckask2f","expires_
in":36000,"refresh_token":"epecdr7zzfjxwwfu35395dx9","scope":null,"user_
context":"testUser1455730448397","uri":null,"extended":null,"state":nul
l}}

```

## Sample Calls through Proxy Endpoint

**i Note:** This section uses the settings from the *Proxy Endpoint Creation in Tethered* section in [OAuth Authenticator Service Configuration](#).

### Create access token

```

curl -v -d '{"id":1,"method":"oauth2.createAccessToken","params":
{"service_key":"m9f5kyzfjjrssh6d5sfkbuw3","client":{"client_

```

```
id":"yq7chp6ufkeea587gp6wqa6x","client_secret":"Azn4Xa"},"token_data":
{"grant_type":"implicit"},"uri":{"redirect_
uri":"https://some.com/cb"},"user_
context":"testUser1455730448397"},"jsonrpc":"2.0"}' -H 'Host:
chainsproxy.api.mashery.com' -H 'Content-Type: application/json'
'http://192.168.99.100:80/v2/oauth/?api_key=8qavf83pg2uzgndtxz2w4dg3'
* Trying 192.168.99.100...
* Connected to 192.168.99.100 (192.168.99.100) port 80 (#0)
> POST /oauth/auth/v2/?api_key=8qavf83pg2uzgndtxz2w4dg3 HTTP/1.1
> Host: chainsproxy.api.mashery.com
> User-Agent: curl/7.43.0
> Accept: */*
> Content-Type: application/json
> Content-Length: 310
>
* upload completely sent off: 310 out of 310 bytes
< HTTP/1.1 200 OK
< X-Mashery-Responder: 8abfe284898c
< Content-Type: application/json; charset=UTF-8
< Cache-Control: no-store
< Content-Length: 231
< Date: Tue, 30 Oct 2018 00:38:26 GMT
<
* Connection #0 to host 192.168.99.100 left intact
{"jsonrpc":"2.0","id":1,"result":{"token_type":"bearer","return_
type":"json","access_token":"unrct8sweny8h3b3kw39wab","expires_
in":36000,"scope":null,"user_
context":"testUser1455730448397","uri":null,"extended":null,"state":nul
l}}
```

## Create authorization code

```
curl -v -d '{ "id":1,"method":"oauth2.createAuthorizationCode","params":
{"service_key":"m9f5kyzfjjrssz6d5sfkbuw3","client":{"client_
id":"yq7chp6ufkeea587gp6wqa6x"},"response_type":"code","uri":{"redirect_
uri":"https://some.com/cb"},"user_context":
"testUser1455730448397"},"jsonrpc":"2.0"}' -H 'Host:
chainsproxy.api.mashery.com' -H 'Content-Type: application/json'
'http://192.168.99.100:80/v2/oauth/?api_key=8qavf83pg2uzgndtxz2w4dg3'
* Trying 192.168.99.100...
* Connected to 192.168.99.100 (192.168.99.100) port 80 (#0)
> POST /oauth/auth/v2/?api_key=8qavf83pg2uzgndtxz2w4dg3 HTTP/1.1
> Host: chainsproxy.api.mashery.com
> User-Agent: curl/7.43.0
> Accept: */*
> Content-Type: application/json
```

```
> Content-Length: 277
>
* upload completely sent off: 277 out of 277 bytes
< HTTP/1.1 200 OK
< X-Mashery-Responder: 8abfe284898c
< Content-Type: application/json; charset=UTF-8
< Cache-Control: no-store
< Content-Length: 168
< Date: Tue, 30 Oct 2018 00:42:12 GMT
<
* Connection #0 to host 192.168.99.100 left intact
{"jsonrpc":"2.0","id":1,"result":
{"code":"p2kjsmn8mw2xb3d2ksryxk6z","uri":{"redirect_
uri":"https://some.com/cb?code=p2kjsmn8mw2xb3d2ksryxk6z","state":""}}, "e
rror":null}
```

### Create access token with authorization code

```
curl -v -d '{ "id":1,"method":"oauth2.createAuthorizationCode","params":
{"service_key":"m9f5kyzfjjrssz6d5sfkbuw3","client":{"client_
id":"yq7chp6ufkeea587gp6wqa6x"},"response_type":"code","uri":{"redirect_
uri":"https://some.com/cb"},"user_context":
"testUser1455730448397"},"jsonrpc":"2.0"}' -H 'Host:
chainsproxy.api.mashery.com' -H 'Content-Type: application/json'
'http://192.168.99.100:80/v2/oauth/?api_key=8qavf83pg2uzgndtxz2w4dg3'
* Trying 192.168.99.100...
* Connected to 192.168.99.100 (192.168.99.100) port 80 (#0)
> POST /oauth/auth/v2/?api_key=8qavf83pg2uzgndtxz2w4dg3 HTTP/1.1
> Host: chainsproxy.api.mashery.com
> User-Agent: curl/7.43.0
> Accept: */*
> Content-Type: application/json
> Content-Length: 277
>
* upload completely sent off: 277 out of 277 bytes
< HTTP/1.1 200 OK
< X-Mashery-Responder: 8abfe284898c
< Content-Type: application/json; charset=UTF-8
< Cache-Control: no-store
< Content-Length: 168
< Date: Tue, 30 Oct 2018 00:42:12 GMT
<
* Connection #0 to host 192.168.99.100 left intact
{"jsonrpc":"2.0","id":1,"result":
{"code":"p2kjsmn8mw2xb3d2ksryxk6z","uri":{"redirect_
```

```
uri":"https://some.com/cb?code=p2kjsmn8mw2xb3d2ksryxk6z","state":""}},"error":null}
```

## Update access token

```
curl -v -d '{"id":1,"method":"oauth2.updateAccessToken","params":
{"service_key":"m9f5kyzfjjrssz6d5sfkbuw3","client":{"client_
id":"yq7chp6ufkeea587gp6wqa6x","client_secret":"Azn4Xa"},"access_
token":"rjzb2k26cgv7tufe2j8cs7rm","user_
context":"testUser1455730448397", "expires_in":600},"jsonrpc":"2.0"}' -H
'Host: chainsproxy.api.mashery.com' -H 'Content-Type: application/json'
'http://192.168.99.100:80/v2/oauth/?api_key=8qavf83pg2uzgndtxz2w4dg3'
* Trying 192.168.99.100...
* Connected to 192.168.99.100 (192.168.99.100) port 80 (#0)
> POST /oauth/auth/v2/?api_key=8qavf83pg2uzgndtxz2w4dg3 HTTP/1.1
> Host: chainsproxy.api.mashery.com
> User-Agent: curl/7.43.0
> Accept: */*
> Content-Type: application/json
> Content-Length: 287
>
* upload completely sent off: 287 out of 287 bytes
< HTTP/1.1 200 OK
< X-Mashery-Responder: 8abfe284898c
< Content-Type: application/json; charset=UTF-8
< Cache-Control: no-store
< Content-Length: 322
< Date: Tue, 30 Oct 2018 01:01:50 GMT
<
* Connection #0 to host 192.168.99.100 left intact
{"jsonrpc":"2.0","id":1,"result":{"token_type":"bearer","access_
token":"rjzb2k26cgv7tufe2j8cs7rm","expires":"2018-10-
30T11:01:50Z","refresh_token_expires":1540864158505,"scope":null,"user_
context":"testUser1455730448397","uri":null,"grant_type":"authorization_
code","client_id":"yq7chp6ufkeea587gp6wqa6x","extended":null}}
```

## Fetch access token

```
curl -v -d '{"id": 1,"jsonrpc":"2.0","method":
"oauth2.fetchAccessToken","params": {"service_
key":"m9f5kyzfjjrssz6d5sfkbuw3","access_token":
"rjzb2k26cgv7tufe2j8cs7rm"}}' -H 'Host: chainsproxy.api.mashery.com' -H
'Content-Type: application/json'
'http://192.168.99.100:80/v2/oauth/?api_key=8qavf83pg2uzgndtxz2w4dg3'
* Trying 192.168.99.100...
```

```

* Connected to 192.168.99.100 (192.168.99.100) port 80 (#0)
> POST /oauth/auth/v2/?api_key=8qavf83pg2uzgndtxz2w4dg3 HTTP/1.1
> Host: chainsproxy.api.mashery.com
> User-Agent: curl/7.43.0
> Accept: */*
> Content-Type: application/json
> Content-Length: 157
>
* upload completely sent off: 157 out of 157 bytes
< HTTP/1.1 200 OK
< X-Mashery-Responder: 8abfe284898c
< Content-Type: application/json; charset=UTF-8
< Cache-Control: no-store
< Content-Length: 242
< Date: Tue, 30 Oct 2018 01:05:25 GMT
<
* Connection #0 to host 192.168.99.100 left intact
{"jsonrpc":"2.0","id":1,"result":{"token_type":"bearer","expires":"2018-10-30T11:01:50Z","scope":null,"user_context":"testUser1455730448397","uri":null,"grant_type":"authorization_code","client_id":"yq7chp6ufkeea587gp6wqa6x","extended":null}}

```

## Fetch application

```

curl -v -d '{"id":1,"jsonrpc":"2.0","method":"oauth2.fetchApplication","params":{"service_key":"m9f5kyzfjrrsz6d5sfkbuw3","client":{"client_id":"yq7chp6ufkeea587gp6wqa6x","client_secret":"Azn4Xa"}}}' -H 'Host: chainsproxy.api.mashery.com' -H 'Content-Type: application/json' 'http://192.168.99.100:80/v2/oauth/?api_key=8qavf83pg2uzgndtxz2w4dg3'
* Trying 192.168.99.100...
* Connected to 192.168.99.100 (192.168.99.100) port 80 (#0)
> POST /oauth/auth/v2/?api_key=8qavf83pg2uzgndtxz2w4dg3 HTTP/1.1
> Host: chainsproxy.api.mashery.com
> User-Agent: curl/7.43.0
> Accept: */*
> Content-Type: application/json
> Content-Length: 187
>
* upload completely sent off: 187 out of 187 bytes
< HTTP/1.1 200 OK
< X-Mashery-Responder: 8abfe284898c
< Content-Type: application/json; charset=UTF-8
< Cache-Control: no-store
< Content-Length: 82
< Date: Tue, 30 Oct 2018 01:10:58 GMT

```



```
<
* Connection #0 to host 192.168.99.100 left intact
{"jsonrpc":"2.0","id":1,"result":
{"id":251547,"name":"TestOAuthRegisterCallBack"}}
```

## Fetch user application

```
curl -v -d '{ "id":1, "method":"oauth2.fetchUserApplications",
"params":{"service_key":"m9f5kyzfjjrssz6d5sfkbuw3", "user_
context":"testUser1455730448397" }, "jsonrpc":"2.0"}' -H 'Host:
chainsproxy.api.mashery.com' -H 'Content-Type: application/json'
'http://192.168.99.100:80/v2/oauth/?api_key=8qavf83pg2uzgndtxz2w4dg3'
* Trying 192.168.99.100...
* Connected to 192.168.99.100 (192.168.99.100) port 80 (#0)
> POST /oauth/auth/v2/?api_key=8qavf83pg2uzgndtxz2w4dg3 HTTP/1.1
> Host: chainsproxy.api.mashery.com
> User-Agent: curl/7.43.0
> Accept: */*
> Content-Type: application/json
> Content-Length: 162
>
* upload completely sent off: 162 out of 162 bytes
< HTTP/1.1 200 OK
< X-Mashery-Responder: 8abfe284898c
< Content-Type: application/json; charset=UTF-8
< Cache-Control: no-store
< Content-Length: 195
< Date: Tue, 30 Oct 2018 01:12:32 GMT
<
* Connection #0 to host 192.168.99.100 left intact
{"jsonrpc":"2.0","id":1,"result":
[{"id":251547,"name":"TestOAuthRegisterCallBack","access_tokens":
["rjzb2k26cgv7tufe2j8cs7rm","unrct8sweny8h3b3kw39wab"],"client_
id":"yq7chp6ufkeea587gp6wqa6x"}]}
```

## Revoke access token

```
curl -v -d '{ "id":1, "method":"oauth2.revokeAccessToken", "params":{"
service_key":"m9f5kyzfjjrssz6d5sfkbuw3", "client": { "client_
id":"yq7chp6ufkeea587gp6wqa6x", "client_secret":"Azn4Xa" }, "access_
token":"rjzb2k26cgv7tufe2j8cs7rm" }, "jsonrpc":"2.0"}' -H 'Host:
chainsproxy.api.mashery.com' -H 'Content-Type: application/json'
'http://192.168.99.100:80/v2/oauth/?api_key=8qavf83pg2uzgndtxz2w4dg3'
* Trying 192.168.99.100...
* Connected to 192.168.99.100 (192.168.99.100) port 80 (#0)
```

```

> POST /oauth/auth/v2/?api_key=8qavf83pg2uzgndtxz2w4dg3 HTTP/1.1
> Host: chainsproxy.api.mashery.com
> User-Agent: curl/7.43.0
> Accept: */*
> Content-Type: application/json
> Content-Length: 240
>
* upload completely sent off: 240 out of 240 bytes
< HTTP/1.1 200 OK
< X-Mashery-Responder: 8abfe284898c
< Content-Type: application/json; charset=UTF-8
< Cache-Control: no-store
< Content-Length: 38
< Date: Tue, 30 Oct 2018 01:17:10 GMT
<
* Connection #0 to host 192.168.99.100 left intact
{"jsonrpc":"2.0","id":1,"result":true}

```

## Revoke user application

```

curl -v -d '{ "id":1, "method":"oauth2.revokeUserApplication",
"params":{"service_key":"m9f5kyzfjjrssz6d5sfkbw3", "client": {
"client_id":"yq7chp6ufkeea587gp6wqa6x", "client_secret":"Azn4Xa" },
"user_context":"testUser1455730448397" }, "jsonrpc":"2.0"}' -H 'Host:
chainsproxy.api.mashery.com' -H 'Content-Type: application/json'
'http://192.168.99.100:80/v2/oauth/?api_key=8qavf83pg2uzgndtxz2w4dg3'
* Trying 192.168.99.100...
* Connected to 192.168.99.100 (192.168.99.100) port 80 (#0)
> POST /oauth/auth/v2/?api_key=8qavf83pg2uzgndtxz2w4dg3 HTTP/1.1
> Host: chainsproxy.api.mashery.com
> User-Agent: curl/7.43.0
> Accept: */*
> Content-Type: application/json
> Content-Length: 241
>
* upload completely sent off: 241 out of 241 bytes
< HTTP/1.1 500 Server Error
< X-Mashery-Responder: 8abfe284898c
< Content-Type: application/json; charset=UTF-8
< Cache-Control: no-store
< Content-Length: 81
< Date: Tue, 30 Oct 2018 16:41:56 GMT
<
* Connection #0 to host 192.168.99.100 left intact
{"jsonrpc":"2.0","id":1,"error":{"message":"Internal Server
Error","code":-2001}}

```

## Refresh token

```

curl -k -v -d '{"id":1,"method":"oauth2.createAccessToken","params":
{"service_key":"m9f5kyzfjjrssz6d5sfkbuw3","client":{"client_
id":"yq7chp6ufkeea587gp6wqa6x","client_secret":"Azn4Xa"},"token_data":
{"grant_type":"refresh_token","refresh_
token":"8f9qy6jrw3r23j2r6wqwp2jh"},"uri":{"redirect_
uri":"https://some.com/cb"},"user_context":null},"jsonrpc":"2.0"}' -H
'Host: chainsproxy.api.mashery.com' -H 'Content-Type: application/json'
'http://192.168.99.100:80/v2/oauth/?api_key=8qavf83pg2uzgndtxz2w4dg3'
* Trying 192.168.99.100...
* Connected to 192.168.99.100 (192.168.99.100) port 80 (#0)
> POST /oauth/auth/v2/?api_key=8qavf83pg2uzgndtxz2w4dg3 HTTP/1.1
> Host: chainsproxy.api.mashery.com
> User-Agent: curl/7.43.0
> Accept: */*
> Content-Type: application/json
> Content-Length: 339
>
* upload completely sent off: 339 out of 339 bytes
< HTTP/1.1 200 OK
< X-Mashery-Responder: 8abfe284898c
< Content-Type: application/json;charset=UTF-8
< Cache-Control: no-store
< Content-Length: 274
< Date: Tue, 30 Oct 2018 16:55:36 GMT
<
* Connection #0 to host 192.168.99.100 left intact
{"jsonrpc":"2.0","id":1,"result":{"token_type":"bearer","return_
type":"json","access_token":"79rkkkxd4rk2zzbrqj4z93vb","expires_
in":3600,"refresh_token":"7db67za3xvdg99nc7gnctkqw","scope":null,"user_
context":"testUser1455730448397","uri":null,"extended":null,"state":nul
l}}

```

## Securing Config UI and Dev Portal in Untethered Setup

The Config UI and Dev Portal are UI components of TIBCO Cloud™ API Management - Local Edition. They can be secured using LDAP and SAML. This section describes how to setup LDAP and SAML server and configure it with Local Edition. Before moving to Local Edition, we have to setup SAML and LDAP server on an instance.

# Configuring LDAP and SAML with Local Edition

The following section provides details to configure [LDAP](#) and [SAML](#) with Local Edition.

## Configuring LDAP with Local Edition

### Procedure

1. Launch Configuration Manager.

```
https://<ip_of_cm_service>:8443/admin
```

2. Navigate to **Members**.
3. Create a new user name LDAP
4. Add the appropriate role to the LDAP user by navigating to **Portal Access Group** tab. The LDAP-Authenticated users inherit the roles and permissions of this LDAP user.
5. Navigate to **Home > Zone Settings** and under **LDAP settings** input the following:

- **LDAP Enabled:** Checked (Enabled)
- **LDAP Login Button Text:** LDAP Login(or any other text you want on the **LDAP Login** button)
- **LDAP URL:** URL of the LDAP Server. This server should be accessible from the Local Edition Cluster. For example,

```
LDAP URL: ldap:// 34. 67.34.161.1234
```

- **Bind Username and Password:** This is used to connect to the LDAP Server. For example,

```
Bind Username: uid=john,ou=People,dc=nodomain  
Bind Password: johnldap
```

- **Search Filter:** Filter to search for the user during authentication. Standard LDAP Filter syntax is supported. For example,

Search Filter: (&(objectclass=inetOrgPerson)(uid=%s))

**uid** can be replaced with any unique identifier. %s represents the username used on the login page. A minimal filter should contain (**<unique\_identifier>=%s**)

- **Base DN:** The Base DN is the starting point a LDAP Server uses when searching for users authentication within the directory.

Base DN: dc=example-domain,dc=com

- **Start TLS:** Checked (Enabled)
- **Skip Verify:** Checked (Enabled)

LDAP Settings

☒ **LDAP Enabled**

Enable LDAP

**LDAP Login Button Text**

The text on the LDAP login button.

**LDAP URL**

The LDAP URL, starts with 'ldap://', 'ldaps://', or 'ldapi://'.

**Bind Username**

The username to use for looking up users.

**Bind Password**

The password to use for looking up users.

**Search Filter**

Example: '(&(objectClass=organizationalPerson)(uid=%s))'

**Base DN**

Example: 'ou=People,dc=nodomain'

☒ **Start TLS**

Use StartTLS with 'ldap://'.

☒ **Skip Verify**

Don't verify the certificate.

Test Status:

LDAP test start....  
Connecting to LDAP server  
Connected to LDAP server  
Switching to TLS  
Successfully connected with TLS  
Binding to authentication user  
Successful user bind  
Success

## 6. Click **Save and Test LDAP**.

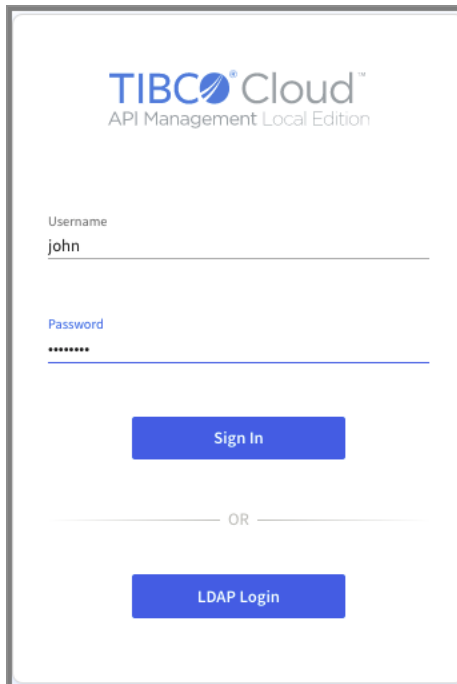
On successful configuration the test status must be :

```
LDAP test start....  
Connecting to LDAP server  
Connected to LDAP server  
Switching to TLS  
Successfully connected with TLS  
Binding to authentication user  
Successful user bind  
Success
```

## Login with LDAP User

The following section provides details to login with a LDAP user.

1. Navigate to the Login page, [https://<ip\\_of\\_cm\\_service>:8443/admin](https://<ip_of_cm_service>:8443/admin)
2. Provide the LDAP Username and password and click the **Sign In** button.



The screenshot displays the login interface for TIBCO Cloud API Management Local Edition. At the top, the logo and product name are visible. Below, there are two input fields: 'Username' containing the text 'john' and 'Password' which is masked with asterisks. A blue 'Sign In' button is positioned below the password field. Underneath this button is a horizontal line with the text 'OR' in the center. At the bottom of the form is a blue 'LDAP Login' button.

On successful authentication, you should be signed-in in the Configuration Manager.

# Configuring SAML with Local Edition

## Procedure

1. Generate SAML certificate and key on SAML Server. Transfer this key and certificate on to local machine.

```
openssl req -x509 -newkey rsa:2048 -keyout myservice.key -out  
myservice.cert -days 365 -nodes -subj "/CN=myservice.example.com"
```

2. Launch the configuration manager and navigate to **Members**.

```
https://<ip_of_cm_service>:8443/admin
```

3. Create a new user as SAML and in the **Portal Access Group** tab, input the Administrator role to the saml user.
4. Click **Home > Zone Settings > SAML settings** and input the following:
  - SAML Enabled: Checked
  - SAML Login Button Text: SAML Login. This can be any other text you want on the SAML Login Button.
  - Metadata URL: http://<saml\_server\_ip>:8000/metadata. This is the IP of VM or instance where SAML server is running.
  - Root URL: https://<cm\_svc\_ip>:8443. Here, cm\_svc\_ip is the IP of cm service that is used to launch the configuration manager.
  - Force Authentication: Checked
  - SAML Certificate: Browse and select the certificate generated in step 1.
  - SAML Key: Browse and select the key generated in step 1.

SAML Settings

☒ **SAML Enabled**
Enable SAML

**SAML Login Button Text**

SAML Login

The text on the SAML login button.

**Metadata URL**

http://<saml\_server\_ip>:8000/metadata

The metadata URL of the SAML identity provider.

**Root URL**

https://<cm\_svc\_ip>:8443

The root URL of the SAML service provider.

☐ **Force Authentication**
Force the SAML identity provider to re-authenticate.

**SAML Certificate**

Choose SAML Certificate

Browse

The certificate to use with SAML authentication.

**SAML Key**

Choose SAML Key

Browse

The key to use with SAML authentication.

Save And Test SAML

##### 5. Click **Save and Test SAML**.

On successful configuration the test status must be :

```

SAML test start....
Loading certificate
Successfully loaded certificate
Parsing certificate
Successfully parsed certificate
Parsing metadata url
Successfully parsed metadata url
Parsing root url
Successfully parsed root url
Creating SAML service
Successfully created SAML service

```

### What to do next

On successful configuration download the `metadata.xml` file by clicking on the metadata link below the test status box.

Load the metadata file to SAML server.



```
curl -v -T metadata.xml http://<saml_server_ip>:8000/services/test
```

```
curl -v -T /Users/<user_name>/Downloads/metadata.xml
http://xx.xx.xx.xxx:8000/services/test
* Trying xx.xx.xx.xxx...
* TCP_NODELAY set
* Connected to xx.xx.xx.xxx (xx.xx.xx.xxx) port 8000 (#0)
> PUT /services/test HTTP/1.1
> Host: xx.xx.xx.xxx:8000
> User-Agent: curl/7.54.0
> Accept: */*
> Content-Length: 3795
> Expect: 100-continue
>
< HTTP/1.1 100 Continue
* We are completely uploaded and fine
< HTTP/1.1 204 No Content
< Date: Thu, 10 Dec 2020 23:49:21 GMT
<
* Connection #0 to host xx.xx.xx.xxx left intact
```


## Logging in the Configuration Manager with LDAP and SAML

### Procedure

1. Launch the configuration manager.

```
https://<ip_of_cm_service>:8443/admin
```

2. Choose from the any of the three login types. The **Login** option is for default admin login or login credentials for members created using the configuration manager.



Username

Password

[Sign In](#)

OR

[LDAP Sign In](#)

OR

[SAML Sign In](#)

**i** **Note:** LDAP and SAML login credential must be different.

# TIBCO Documentation and Support Services

---

For information about this product, you can read the documentation, contact TIBCO Support, and join TIBCO Community.

## How to Access TIBCO Documentation

Documentation for TIBCO products is available on the [TIBCO Product Documentation](#) website, mainly in HTML and PDF formats.

The [TIBCO Product Documentation](#) website is updated frequently and is more current than any other documentation included with the product.

## Product-Specific Documentation

The following documentation for this product is available on the [TIBCO Cloud™ API Management - Local Edition Product Documentation](#) page:

- *TIBCO Cloud™ API Management - Local Edition Release Notes*
- *TIBCO Cloud™ API Management - Local Edition Installation and Configuration Guide*
- *TIBCO Cloud™ API Management - Local Edition Migration and Upgrade Guide*
- *TIBCO Cloud™ API Management - Local Edition SDK Guide*
- *TIBCO Cloud™ API Management - Local Edition Reporting Services Guide*
- *TIBCO Cloud™ API Management - Local Edition Cluster Design Guide*
- *TIBCO Cloud™ API Management - Local Edition Security Guide*

## Other TIBCO Product Documentation

When working with Local Edition, you may find it useful to read the documentation of the following TIBCO product:

- [TIBCO Cloud API Management™](#): a modern API platform for digital business that covers the entire API lifecycle.

## How to Contact TIBCO Support

Get an overview of [TIBCO Support](#). You can contact TIBCO Support in the following ways:

- For accessing the Support Knowledge Base and getting personalized content about products you are interested in, visit the [TIBCO Support](#) website.
- For creating a Support case, you must have a valid maintenance or support contract with TIBCO. You also need a user name and password to log in to [TIBCO Support](#) website. If you do not have a user name, you can request one by clicking **Register** on the website.

## How to Join TIBCO Community

TIBCO Community is the official channel for TIBCO customers, partners, and employee subject matter experts to share and access their collective experience. TIBCO Community offers access to Q&A forums, product wikis, and best practices. It also offers access to extensions, adapters, solution accelerators, and tools that extend and enable customers to gain full value from TIBCO products. In addition, users can submit and vote on feature requests from within the [TIBCO Ideas Portal](#). For a free registration, go to [TIBCO Community](#).

# Legal and Third-Party Notices

---

SOME TIBCO SOFTWARE EMBEDS OR BUNDLES OTHER TIBCO SOFTWARE. USE OF SUCH EMBEDDED OR BUNDLED TIBCO SOFTWARE IS SOLELY TO ENABLE THE FUNCTIONALITY (OR PROVIDE LIMITED ADD-ON FUNCTIONALITY) OF THE LICENSED TIBCO SOFTWARE. THE EMBEDDED OR BUNDLED SOFTWARE IS NOT LICENSED TO BE USED OR ACCESSED BY ANY OTHER TIBCO SOFTWARE OR FOR ANY OTHER PURPOSE.

USE OF TIBCO SOFTWARE AND THIS DOCUMENT IS SUBJECT TO THE TERMS AND CONDITIONS OF A LICENSE AGREEMENT FOUND IN EITHER A SEPARATELY EXECUTED SOFTWARE LICENSE AGREEMENT, OR, IF THERE IS NO SUCH SEPARATE AGREEMENT, THE CLICKWRAP END USER LICENSE AGREEMENT WHICH IS DISPLAYED DURING DOWNLOAD OR INSTALLATION OF THE SOFTWARE (AND WHICH IS DUPLICATED IN THE LICENSE FILE) OR IF THERE IS NO SUCH SOFTWARE LICENSE AGREEMENT OR CLICKWRAP END USER LICENSE AGREEMENT, THE LICENSE(S) LOCATED IN THE “LICENSE” FILE(S) OF THE SOFTWARE. USE OF THIS DOCUMENT IS SUBJECT TO THOSE TERMS AND CONDITIONS, AND YOUR USE HEREOF SHALL CONSTITUTE ACCEPTANCE OF AND AN AGREEMENT TO BE BOUND BY THE SAME.

This document is subject to U.S. and international copyright laws and treaties. No part of this document may be reproduced in any form without the written authorization of Cloud Software Group, Inc.

TIBCO, the TIBCO logo, the TIBCO O logo, and the TIBCO Cloud logo are either registered trademarks or trademarks of Cloud Software Group, Inc. in the United States and/or other countries.

Additional third-party information is available on [TIBCO Community](#).

This document includes fonts that are licensed under the SIL Open Font License, Version 1.1, which is available at: <https://scripts.sil.org/OFL>

Copyright (c) Paul D. Hunt, with Reserved Font Name Source Sans Pro and Source Code Pro.

All other product and company names and marks mentioned in this document are the property of their respective owners and are mentioned for identification purposes only.

This software may be available on multiple operating systems. However, not all operating system platforms for a specific software version are released at the same time. See the readme file for the availability of this software version on a specific operating system platform.

THIS DOCUMENT IS PROVIDED “AS IS” WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT.

THIS DOCUMENT COULD INCLUDE TECHNICAL INACCURACIES OR TYPOGRAPHICAL ERRORS. CHANGES ARE PERIODICALLY ADDED TO THE INFORMATION HEREIN; THESE CHANGES WILL BE INCORPORATED IN NEW EDITIONS OF THIS DOCUMENT. CLOUD SOFTWARE GROUP, INC. MAY MAKE IMPROVEMENTS AND/OR CHANGES IN THE PRODUCT(S) AND/OR THE PROGRAM(S) DESCRIBED IN THIS DOCUMENT AT ANY TIME.

THE CONTENTS OF THIS DOCUMENT MAY BE MODIFIED AND/OR QUALIFIED, DIRECTLY OR INDIRECTLY, BY OTHER DOCUMENTATION WHICH ACCOMPANIES THIS SOFTWARE, INCLUDING BUT NOT LIMITED TO ANY RELEASE NOTES AND "READ ME" FILES.

This and other products of Cloud Software Group, Inc. may be covered by registered patents. Please refer to TIBCO's Virtual Patent Marking document (<https://www.tibco.com/patents>) for details.

Copyright © 2004-2023. Cloud Software Group, Inc. All Rights Reserved.