



Boomi Cloud™ API Management - Local Edition

Installation and Configuration Guide

Version 5.6.2 | November 2024

Contents

Contents	2
Boomi Cloud™ API Management - Local Edition Overview	9
Local Edition Topology	10
Local Edition Component Interactions	11
Quick Start	13
Local Edition Components	25
Components Overview	25
NoSQL	26
Registry	28
Log Service	30
Accessing QPM Data by Cluster for Untethered Deployments	34
SQL DB	36
Onprem Loader	41
Cache	44
Memcached Loader	49
Traffic Manager	51
TLS Server Security	55
TLS Client Security	56
Cluster Manager	57
Cluster Manager CLI	58
API Management - Local Edition API	96
Configuration Manager Overview	98
Developer Portal Overview	98
Local Edition Installer	99
Installer Overview	99

HTTP Proxy Support	99
HTTP Proxy Support with Authentication	101
Deploying the Installer	102
How to Run the Installer on Windows	103
Setting up HTTP Proxy Authentication in Installer Jenkins	105
Managing the Installer	106
Prebuilding	109
Building and Verifying the Docker Images	113
Building and Installing Local Edition Connectors	117
Prepushing	119
Pushing the Docker Images	123
Deploying the Local Edition Cluster	127
Deploying a Local Edition Cluster	132
Deployment Topologies	132
Docker Container View of a Single Zone	133
Local Edition Cluster in Kubernetes / Docker Swarm - Single Zone	134
Local Edition Cluster in Kubernetes Docker Swarm - Multizone	135
User Content, Templates and Resources	135
User Templates	136
Example	138
FAQ	139
Configuring Properties Common to All Deployments	140
Ports to Open Externally	160
Changing the MySQL Password of a Running Cluster	161
API Debug Key Common to All Deployments	163
Deployment Sequence	166
MySQL Replication Groups	173
Normal State of MySQL Replication Group	173
Abnormal State of MySQL Replication Group	174
Rebuilding MySQL Replication Group	175
Encrypting MySQL Replication Group	178

Installing MySQL Standard (Commercial) Edition in Local Edition	179
Deployments on Amazon AWS	182
Configuration	183
Deployments on Microsoft Azure	185
Configuring	186
Deployments on Google Cloud Platform	188
Configuration	189
Deployment on On-premises	190
Configuration	191
Common Configurations for all Deployment Platforms	193
Deployment on Kubernetes/Openshift	205
Create TML Cluster	214
Validate TML Cluster	215
Deployment on Docker Swarm	216
Setting up Mashery Service Configuration data	219
Validate TML Cluster	225
Making a traffic call	227
Delete TML Cluster	228
Deployment on one Tethered Mode Local Edition 5.5.x Cluster across Multiple Kubernetes or OpenShift Clusters	228
Configuration Manager	235
Managing Configuration Data	235
Configuring Zone Settings	237
Configuring API Definitions and Endpoints	244
Configuring API Packages and Plans	255
Configuring Apps and Package Keys	259
Configuring API Domains	261
Configuring Members	262
Configuring Roles	264
Configuring API HTTPS Client Profiles	265
Configuring the Content Management System	266

Searching Package Keys	272
Configuring Interactive Documentation	273
Configuring Portal Domains	274
Configure SMTP/Admin Emails	275
Managing Portal Settings	277
Managing Organizations	278
Managing Schedule Maintenance Events	288
LDAP and SAML	289
Configuring LDAP and SAML with Local Edition	289
Logging in the Configuration Manager with LDAP and SAML	294
FAQ and Troubleshooting	295
Developer Portal	298
Getting Started	298
Portal Roles	298
Default Developer Portal Pages	298
Configuring your Developer Portal Pages	300
Changing the Administrator Password	300
Advanced Customizations for your Developer Portal	303
Customizing your CSS	304
Modifying Templates and Layouts	306
Loading Custom Fonts	311
Configuring Developer Portal Settings	313
Including Boomi Cloud API Management Plugins	315
Adding Third-party Scripts, Styles, and Plugins	316
Translations	318
JavaScript APIs	334
Deploying Default Portal Customizations and Assets to Your Clusters	371
Phase: Design and Development	371
Phase: Production	374
Managing Logs in a Cluster	375

Viewing Logs	375
Interacting with Loggers	376
Collecting Logs	381
Performing Log Cleanup	384
Enriched Access Logs	388
Exporting Logs to Different Output Channels	392
Recipe for File Destination	393
Recipe for Elasticsearch	394
Recipe for HTTP Endpoints	396
Recipe for Kafka	398
Recipe for TCP Socket	400
Recipe for SYSLOG Socket	402
Recipe for MOM and HTTP	404
Configuring Local Edition to Send Logs to Splunk	405
Recipe for Database	406
Recipe for Forward	408
Integrating with TIBCO Spotfire	409
Preparing TIBCO Spotfire to Consume Access Logs	410
Changing Output Destinations for Different Types of Logs using CLI	412
JSON Configuration Files Format for Different Outputs	437
Local Edition Access Log Sync	441
Payload Logging	445
HTTPS Configuration	460
HTTPS Configuration Overview	460
HTTPS Server Configuration	461
Traffic Manager as an HTTPS Server with Mutual SSL	461
HTTPS Client Configuration	463
HTTPS Endpoint Configuration	463
Testing the HTTPS Client Call	464
Troubleshooting HTTPS Problems	465
HTTPS Mutual Authentication	466

Setting up HTTPS Authentication	466
Creating Mutual Authentication	473
OAuth	476
Configuring OAuth 2.0 API Access	476
OAuth APIs and Authenticator Service	476
Making OAuth 2.0 Calls	497
OAuth API Supported Methods	497
Sample Calls	498
Local Edition 5.x OAuth API Compatibility with Local Edition 4.x	514
OAuth Token Migration	516
Structure of Token Migration Utility	516
Deploying the OAuth Token Migration Utility	517
OAuth Token Migration Configuration Properties	526
Running the OAuth Token Migration Utility	529
Verifying OAuth Token Migration	536
OAuth Token Migration FAQs	537
Troubleshooting OAuth Token Migration Issues	538
Monitoring Local Edition Components	542
Format of Metrics	542
Resource Monitoring	542
Component Metrics Monitoring	551
Traffic Manager Metrics	552
SQL Container Metrics (MySQL DB)	566
Cache Container Metrics (Memcached)	571
NoSQL Container Metrics (Cassandra)	578
Onprem Loader Monitoring	592
Memcached Loader Monitoring	594
Functional Monitoring	598
Recipes for ELK	600
Connectors and Adapters	601

Boomi-provided Connectors and Adapters	603
Verifying the Boomi Cloud API Management Connectors and Adapters	604
FAQs	611
Customer-provided Adapters	613
Build in Jenkins	615
Troubleshooting	617
Boomi References	620

Boomi Cloud™ API Management - Local Edition Overview

Boomi Cloud™ API Management - Local Edition(Local Edition) is a cloud-native API gateway that can be deployed within a customer's datacenter or on cloud platforms.

Local Edition supports two deployment modes: Untethered, without any external dependencies, or hybrid in conjunction with Boomi Cloud™ API Management (CAM). These two deployment modes are mutually exclusive. In hybrid mode, Boomi Cloud™ API Management includes the cloud-hosted Developer Portal, Administration Dashboard, and API Reporting and Analytics. In untethered mode, Local Edition includes a local Developer Portal and Configuration Manager. It also includes the ability to update policies using the Local Edition API. Local Edition Configuration Manager and Local Edition API are no longer mutually incompatible like in previous 5.x versions. In untethered mode, logs from Local Edition can be collected and forwarded for downstream reporting and analytics via other externally-available tools. Local Edition is available to customers for installation in container-based environments.

Local Edition Components include:

- [NoSQL](#)

Each NoSQL component is a Cassandra Node. Multiple nodes can be deployed as a ring in a Data Center that maps to a Local Edition Zone and Kubernetes Zone. Local Edition relies on Cassandra for data availability across Zones in a Local Edition Cluster.

- [Registry](#)

The Registry component is responsible for storing the topology of a Local Edition cluster. Every Local Edition component communicates with the registry so it knows about each component's physical information (host, IP, Container Agent URL), its state, and bindings to resources.

- [Log](#)

The Log service component provides the data collection services for all Local Edition components. It also provides Log Consumer services to be able to send the logs to various destinations for processing and analytics.

- [SQL](#)

The SQL component runs a MySQL Database and stores the configuration data related to services, packages, keys, applications and related information.

- [Cache](#)

The Cache component runs memcached pools and provides in-memory cache for fast access to configuration and OAuth data.

- [Traffic Manager](#)

The Traffic Manager component provides the traffic management and OAuth capabilities.

Local Edition components can be deployed in a single zone or multiple zones to form a cluster of components to address scalability, reliability, and high availability requirements. Local Edition relies on Cassandra for data availability within and across Zones.

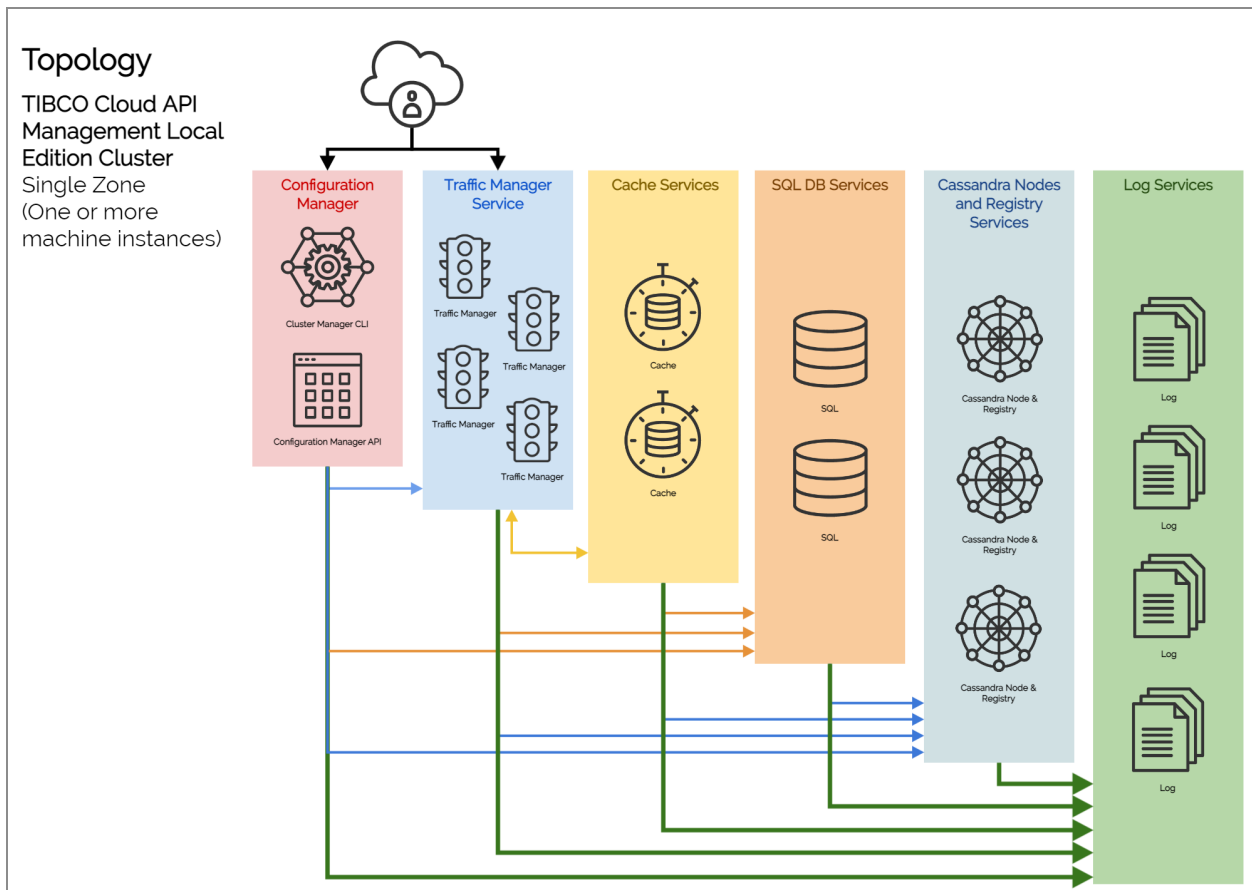
Definition of a Cluster

"Cluster" means a Local Edition installation with all the necessary Local Edition components located at the same physical address (for example, 3307 Hillview Avenue) if being deployed within a customer's premises; or one region, if being deployed in the cloud.

Any installation spanning across multiple regions in the cloud (or multiple physical addresses if within customer's premises) are counted as separate Clusters.

Local Edition Topology

The following image represents the Local Edition topology:

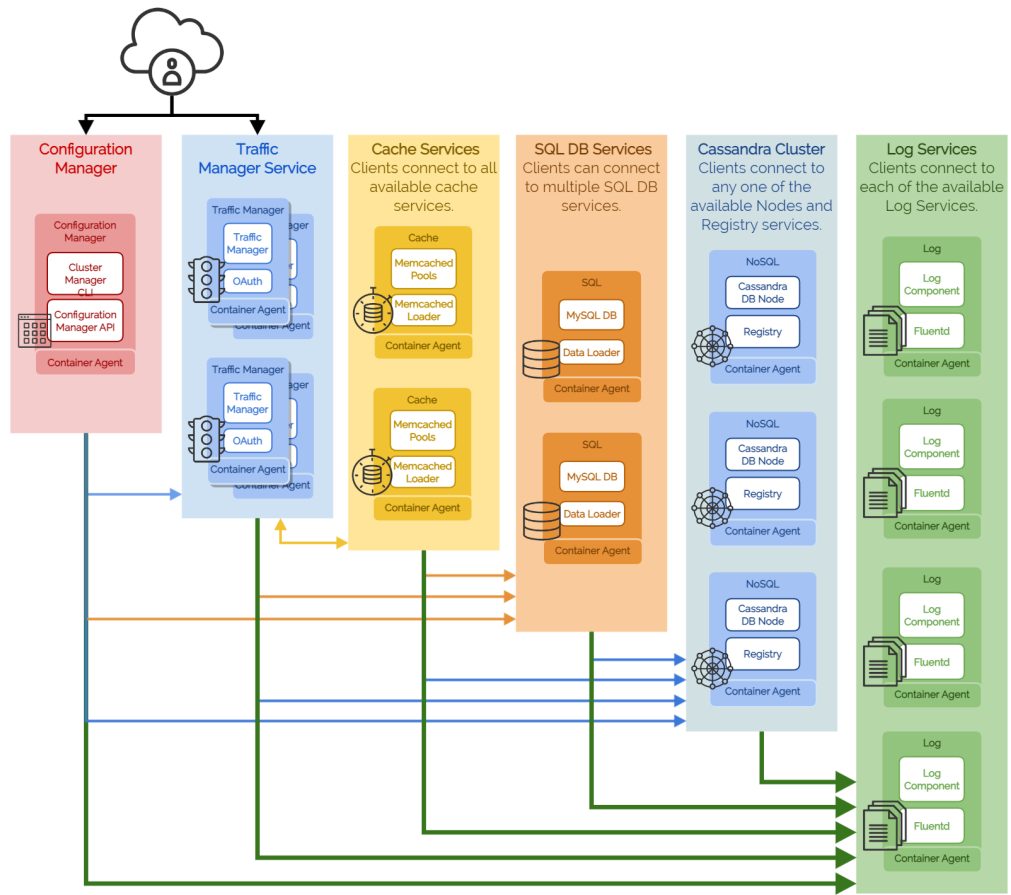


Local Edition Component Interactions

The following image represents the interactions between the Local Edition components:

Component Interactions

TIBCO Cloud API Management Local Edition Cluster
Single Zone
(One or more machine instances)



Quick Start

Overview

Use this quick start section to get your Local Edition Docker images built and deployed with minimal (one container of each component) and default configuration on a single Linux-based or MacOS-based host.

This creates the Local Edition cluster in untethered mode.

Software Prerequisites

- Docker must be installed and running on your machine.

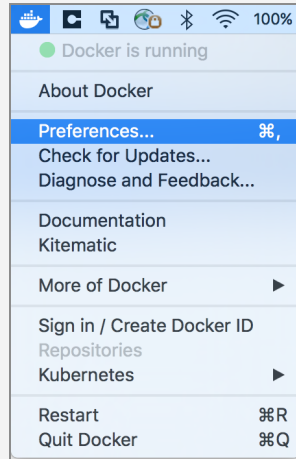


Note: To check if you have Docker installed, run command `docker ps` or `docker info` on a terminal screen to verify it is installed and running. If the command is not found, you need to install Docker. For all platforms, see the [Docker documentation](#).

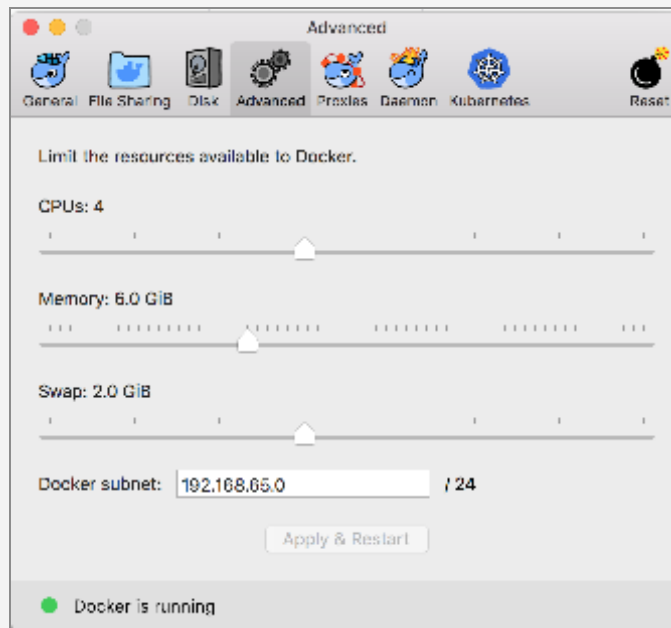
- Docker has been allocated a minimum of 6GB memory for Local Edition cluster and Installer.

Note:

1. Click the **Docker** icon.
2. Click **Preferences**.



3. Go to **Advanced** tab as shown below.



- 'yq' must be installed on the machine.
- You have the Local Edition Installer in tar.gz format (Example : TIB_mash-local_5.2.0.60_cf.tar.gz) on your machine.

Platform Prerequisites

The quick start setup is supported on Linux-based and MacOS-based machines. Quick start uses the local physical machine and not a virtual machine. It sets up a bare-metal swarm cluster of one node (the local physical machine) and one instance of each type of component. It creates a Docker overlay network for the components to communicate over. There are no additional prerequisites for the quick start setup.

Planning

The Local Edition cluster has a single, logical zone, one instance of each type of component, and uses the local machine as the single node. No additional planning is required for the quick start setup.

Configuration

The Local Edition cluster created using the quick start setup uses the untethered mode of configuration. It uses default credentials and does not need any user inputs to configure the properties, sizes or credentials.

Building and Deploying

1. Extract the Installer tar.gz file to your desired location and navigate to the folder `/<Extracted Folder>/quickstart/` on your terminal.
2. For a smooth execution, you should stop any previously running tml-installer container (if any) and remove the volume associated with it. Execute the following commands in sequence. This step can be skipped if you are running tml-installer for the first time.

Sample execution

- a.

```
docker ps -a
```
- b.

```
docker kill <tml-installer container id>
```
- c.

```
docker volume rm -f tml-quickstart_jenkinsvol
```

3. Execute the script `quick-start.sh` with parameters. The script accepts two parameters:

- -b - Set to true to build the images.
- -d - Set to true to create a Local Edition cluster.

Currently, these parameters are mandatory.

Sample execution

```
./quick-start.sh -b true -d true
```

The installer will use default inputs and build the Local Edition images.

The quick start script will then create an overlay Docker network, use swarm to create a single node cluster and proceed to create the simple Local Edition cluster.

If everything goes well, the Local Edition cluster should be ready in around 30-35 minutes (Including Local Edition image building and deployment). At the end of the deployment stage, you may see the message Waiting for TML components to be ready for a long time (10-15 minutes) until everything is active.

4. The [Developer Portal](https://localhost:10443/) can be accessed at <https://localhost:10443/> and [Configuration Manager](https://localhost:10443/admin) can be accessed at <https://localhost:10443/admin>.

i Note: Since the cluster is created with default values, you can use the following default username and password to login into Configuration Manager:

UserName: admin

Password: Ap1Us3rPasswd

The default username and password used to log in to the Configuration Manager can also be found in the `tml_papi_properties.json` file (see the *Platform API Properties* topic in [Configuring Properties Common to all Deployments](#)).

Troubleshooting

If you see the error Error response from daemon: This node is not a swarm manager. Use "docker swarm init" or "docker swarm join" to connect this node to swarm and try again during execution, follow the steps below:

1. Stop/kill the running script.
2. Run `docker swarm init`.

3. Rerun the quick start with parameters:

```
./quick-start.sh -b false -d true
```

Other Use Cases

To only build the image, run the following script:

```
./quick-start.sh -b true -d false
```

If you built the image previously, and now want to run deployment only, run the following script:

```
./quick-start.sh -b false -d true
```

Validating the Local Edition Cluster

Now that the build stage is done, if the deployment is still running, you can check on the progress using following steps:

1. Check the running Docker processes:

```
$ docker ps
```

CONTAINER ID	IMAGE	COMMAND NAMES	CREATED	STATUS
8544fa2e38d4	tml-tm:v5.2.0.1	"/usr/local/bin/tm-e..."	About a minute ago	Up
22/tcp, 80/tcp, 443/tcp, 8000/tcp, 8083-8084/tcp		tmstack_		
tm.1.i40jamp8r5m6x6zoszez17y0h				
45bda152014d	tml-cache:v5.2.0.1	"/usr/local/bin/cach..."	About a minute ago	Up
About a minute	11211-11216/tcp	cachestack_		
cache.1.nylb994ie42d7isjndap3cpvw				
62186c91e18e	tml-sql:v5.2.0.1	"/usr/local/bin/sql-..."	About a minute ago	Up
About a minute	3306/tcp	sqlstack_sql.1.4uk80bcz1dbrbppephki0ikx6r		About
5b22a452dc9f	tml-log:v5.2.0.1	"/usr/local/bin/log-..."	About a minute ago	Up
About a minute	24023/tcp, 24220/tcp, 24224/tcp	logstack_		About
log.1.ue4vb7di5ir8sl55hb8y28fjl				
5a527d7ef2ab	tml-cm:v5.2.0.1	"/usr/local/bin/cm-e..."	2 minutes ago	Up 2
minutes (healthy)		cmstack_tmlcm.1.tj6dbpr1p6zjk4gfh482iwlmh		
365297c18733	tml-nosql:v5.2.0.1	"/usr/local/bin/nosq..."	2 minutes ago	Up 2
minutes (healthy)	7000-7001/tcp, 7199/tcp, 9042/tcp, 9160/tcp	nosqlstack_nosql_		
seed.1.n2ci4fu3i75kc80lq7p49v24x				
5bd04361f7ee	tml-installer:v5.2.0.119	"/sbin/tini -- /usr/..."	26 minutes ago	Up 26

```
minutes      0.0.0.0:8080->8080/tcp, 0.0.0.0:50000->50000/tcp tml-installer
```

2. Log in to the Configuration Manager container and check the components and their states:

```
$ docker exec -ti 5a527d7ef2ab bash
[root@5a527d7ef2ab builder]#
[root@5a527d7ef2ab builder]# alias cm=clustermanager
[root@5a527d7ef2ab builder]# cm ls components
Using cluster [Tibco Mashery Local Reference Cluster]
Using Zone [local]
Component ID          Type      Name      Status    Last Heartbeat Received  Host
Service Port(s)
-----
b38efc8f-bffa-4a68-bcf5-a21370434b78 cache      cache  ACTIVE    Mar 12 2020
08:59:09 +0000 10.0.0.12 11212,11211,11213,11214,11215,11216
f1d75242-4c67-4eb9-a8b3-63195689410c configmanager cm    RESOLVED  Mar 12
2020 08:59:25 +0000 10.0.0.6 7080
11261544-ef93-4d1f-8da9-af71b1593763 logservice log    ACTIVE    Mar 12 2020
08:59:30 +0000 10.0.0.8 24224
66157d26-43b5-40ff-aefe-b1f52aa2f44b nosql      nosql  ACTIVE    Mar 12 2020
08:59:09 +0000 10.0.0.3 9042
fa66dcbb-35af-4fb3-bc87-09382c0f20ef sql        sql    ACTIVE    Mar 12 2020 08:59:34
+0000 10.0.0.10 3306
84792b97-b4ba-4a28-b7f4-1b5cfbd6c76a trafficmanager tm    STANDBY   Mar 12 2020
08:59:28 +0000 10.0.0.14 8080
```

3. Check the status of the overall cluster and see what action is pending:

```
[root@5a527d7ef2ab builder]# cm cluster status
*** Checking the status of Cluster 'Tibco Mashery Local Reference Cluster' ***
*** Checking the status of zone 'local' ***
TML component [66157d26-43b5-40ff-aefe-b1f52aa2f44b] of type [nosql] found with status
[ACTIVE] on Host IP [10.0.0.3]..
TML component [11261544-ef93-4d1f-8da9-af71b1593763] of type [logservice] found with
status [ACTIVE] on Host IP [10.0.0.8]..
TML component [fa66dcbb-35af-4fb3-bc87-09382c0f20ef] of type [sql] found with status
[ACTIVE] on Host IP [10.0.0.10]..
TML component [b38efc8f-bffa-4a68-bcf5-a21370434b78] of type [cache] found with status
[ACTIVE] on Host IP [10.0.0.12]..
TML component [84792b97-b4ba-4a28-b7f4-1b5cfbd6c76a] of type [trafficmanager] found
```

with status [STANDBY] on Host IP [10.0.0.14]. See details below

Component ID	Component Name	Component IP	Component Status
Pending Activity	Error		

84792b97-b4ba-4a28-b7f4-1b5cfbd6c76a	tm	10.0.0.14	Not Ready

- After some time, the cluster status should turn READY. You can check the last failing component for its individual status as follows. In this case, the Traffic Manager is running on 10.0.0.14 IP:

```
[root@5a527d7ef2ab builder]# curl http://10.0.0.14:9080/container/status
{
  "tmgcld" : "84792b97-b4ba-4a28-b7f4-1b5cfbd6c76a",
  "ip" : "10.0.0.14",
  "name" : "tm",
  "pending_activity" : "checking component status ...",
  "status" : "Ready"
}
```

As seen in previous step, the Traffic Manager is in ready status, despite what the pending activity indicates. The pending activity was the last successful activity performed by the container agent.

- If the status indicates error, check the log-service container for corresponding errors.
 - Log in to the logservice container
 - Check the appropriate log files in the /mnt/data directory
 - Check for any errors in the Configuration Manager container. For example, check the /mnt/data/tml-cm directory.

Creating the Cluster Configuration

Follow these steps to create a "Hello World" API example using Configuration Manager.

- Login to Configuration Manager.
- Click on **API Domains** to whitelist your public and internal API domains. Create a new entry for the external and internal domain you plan to use.

Home / API Domains / New

API Domain

Save Cancel

Domain

api.example.com

3. Here, create two entries:

- **api.example.com**: The public domain name that you plan to use.
- **<your local IP address>**: To access your internal Hello World service that runs on your local machine.

4. Return to the main screen and click **API Definitions**. This is where you'll link your public and internal APIs.

API Definition

Home / API Definitions / HelloAPI

HelloAPI

Save Cancel Delete

API Service Key

4rxawqgmbtzbgqcnbeXj6v

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

HelloAPI

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

Modify the description of the API. You can enter purpose, limitations, or other important information about the API.

Version

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

Enter the data for cross-domain policy for API that can be accessed by using Flash.

Robots Policy

Enter the data for robots policy. If no policy information is entered, the default policy allows bots an unrestricted access to the API.

5. Create an API called HelloApi.
6. Click on **Endpoints** on the left-hand pane. Now you can link the public and internal domains that you whitelisted earlier and specify the API path details.

API Definition

Security

Performance Acceleration

Portal Access Groups

Endpoints

Error Sets

Home / API Definitions / API Definition / Endpoints / New

Endpoint

Save Cancel

Name

My Endpoint

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

☐ Require Mutual TLS (MTLS)

Published Endpoint

Protocol Domain Path

HTTP/HTTPS api.example.com /hello

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

HTTP 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

None

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

The published **Endpoint** will specify the publicly accessible domain and path for your API. In this example, it is api.example.com.

The **Origin Endpoint** contains details about the service to be accessed. For this example, your hello service is used, which runs on your <local ip> on port 9080 with path /hello.

Note: In the **Origin Endpoint**, add port 9080 after your local IP and use path /hello only to access the inbuilt Hello Service.

7. Return to the main screen and click on **API Packages**. For your reference, the API package is called **Hello Package**.

Home / API Packages / New

API Package

Save Cancel

Organization

Area Level

Name

Hello Pacakge

Enter a Package name. This field allows you to modify the Package name.

Description

Describe the Package.

- Click on **Plans** on the left-hand pane and create a new Plan. For your reference, the Plan is called **Hello Plan**.

API Package

Home / API Packages / Hello Package / Plans / Hello Plan

Key Properties

Plans

Plan

Save Cancel

Name

Hello Plan

Enter an API Plan name for future reference. API Plan name is displayed while creating Packages and generating reports.

Description

Describe the API Plan. Enter the purpose, limitations, or other important information about the API Plan.

Status

Active

Plan status is available for internal reference only. It does not have any effect on the Plan or related Applications and Package Keys.

Delete

- Click on **Definitions** to link your previously created endpoint to this package and plan.

Activate the API definition to associate with your package and plan from the list of "Available API Definitions".

Once you activate the API definitions, they will show up as **Activated API Definitions** as seen in the previous screenshot.

- Return to the main screen and click on **Applications**. Create a new Application and select a member for it.

Available API Definitions		Activated API Definitions	
Name	Actions	Name	Actions
HelloApi	Activate	HelloApi	

11. Click **Package Keys** on the left-hand pane and create a new API key for your service.
12. Choose your package and plan. In this example, select **Hello Package** and **Hello Plan**. Ensure the status is changed to **Active**, then click **Save**.

13. When you click on **Package Keys** in the left-hand pane, you should see the API key as **Active**.

API Key	Status	Package - Plan	Throttle	Quota	Quota Period	Updated	Actions
xfsffu6t6tyfkmfvmck2rqfk	active	Hello Package - Hello Plan	2	5000	day	Nov 18 2021 10:15	

14. Once the API key is **Active**, note your API key value. Your endpoint will now flow

through Local Edition and should be available for successful invocation in a maximum of 15 minutes.

15. Test the API key by invoking the curl call:

```
$ curl -H 'Host: api.example.com' 'http://localhost/hello?api_key=bwhm6qasmuwrpd5uzu4h648j'

{"time":"2019-09-09 22:36:02.630+0000","message":"Hello world"}
```



Note: This curl call does not need to be run inside any containers; it can be run from the same location and way you run Docker commands.

Troubleshooting Possible Errors

After creating and activating your API, you may run into the following errors until the data flow is complete. It can take maximum 15 minutes for the API to be active.

- **Error 596:**

```
$ curl -H 'Host: api.example.com' 'http://localhost/hello?api_key=stgqte7fz73jdey54tahpr9c'

<h1>596 Service Not Found</h1>
```

- **Error 403:**

```
$ curl -H 'Host: api.example.com' 'http://localhost/hello?api_key=stgqte7fz73jdey54tahpr9c'

403 Forbidden
```

Deleting the Local Edition Cluster

Run the following command to delete a cluster:

```
delete-quickstart-cluster.sh .
```

Local Edition Components

The following sections describe Local Edition components in detail.

- [Components Overview](#)
- [NoSQL](#)
- [Registry](#)
- [Log](#)
- [SQL](#)
- [Cache](#)
- [Traffic Manager](#)
- [Cluster Manager](#)
- [Local Edition API](#)
- [Configuration Manager](#)
- [Developer Portal](#)

Components Overview

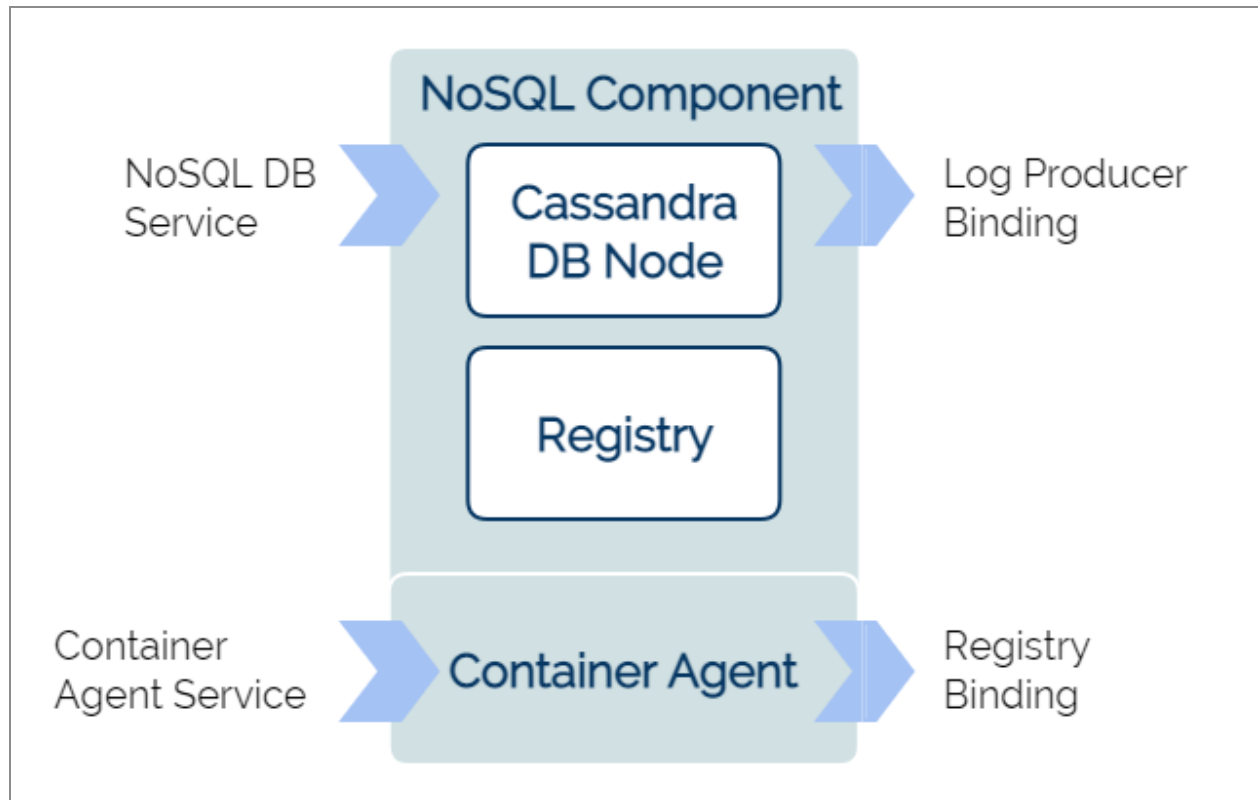
Each Local Edition component has:

- a container agent that provides access to the component and interacts with the registry on its behalf.
- one or more managed processes that are started by the container agent. These provide the services and functionality of a Local Edition component.

The container agent is responsible for:

- starting the managed processes within a Local Edition component.
- interacting with the registry to provide properties, resource bindings and report status on the Local Edition component.

NoSQL



What's Inside

Cassandra Node

Services

NoSQL DB Service

Resource Dependencies

- LogProducer Binding
- Registry Binding

Properties

Name	Default Value	Comment
auto_binding	ON	When turned ON, logservice URLs are auto-discovered and bound to the Local Edition component.
logserviceURL	log-set-0.log-svc.default.svc.cluster.local	Logservice URL to connect to process the logs.
td_agent_bit_buffer_type	filesystem	The back pressure buffer type for logs - store in memory or back up to disk. In most conditions, it would not be required to change the default value; but if high disk i/o is noticed, change the setting. Valid values are memory and filesystem.
td_agent_bit_metrics_buffer_type	filesystem	The back pressure buffer type for metrics - store in memory or back up to disk. In most conditions, it would not be required to change the default value; but if high disk i/o is noticed, change the setting. Valid values are memory and filesystem.

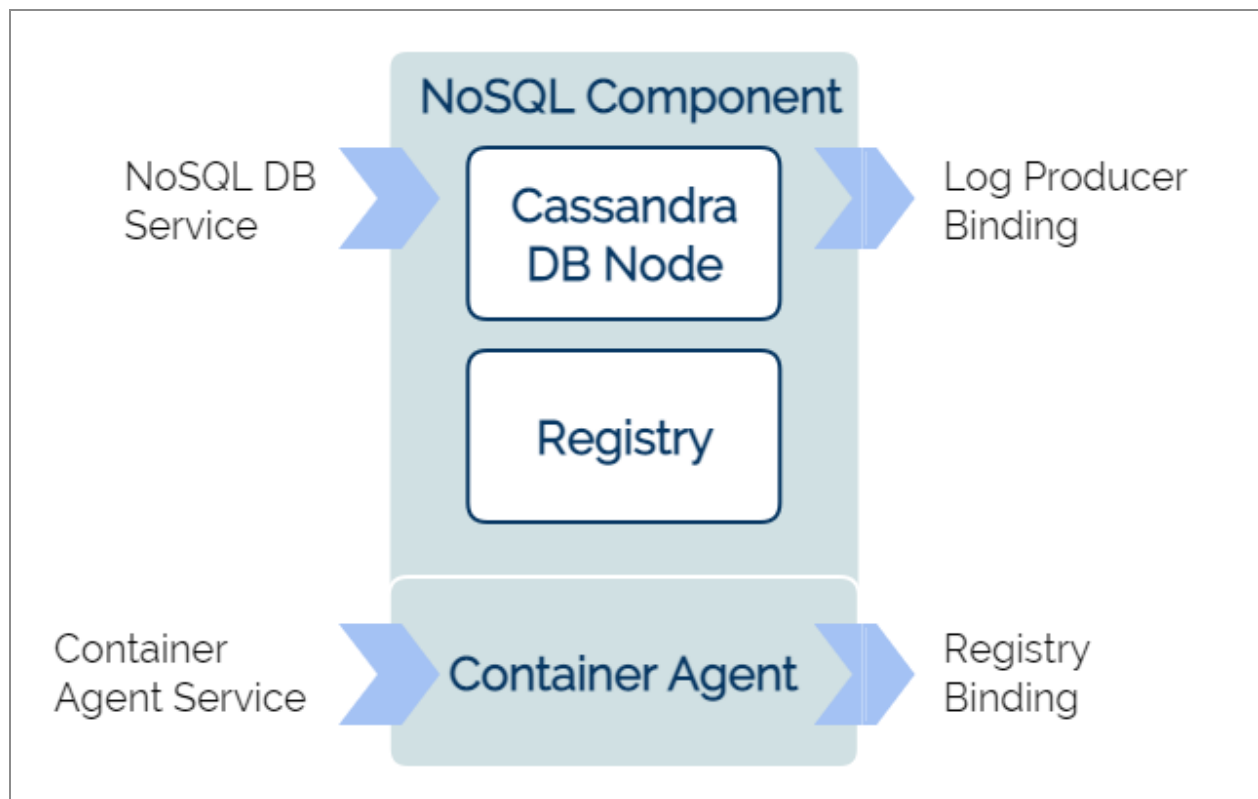
i Note: `td_agent_bit_buffer_type` and `td_agent_bit_metrics_buffer_type` properties can be set at deployment time or can be modified using the Cluster Manager CLI.

For example:

```
cm ls components
```

```
clustermanager set config --componentType <component type from ls components> --  
componentId <component id from ls components> --prop td_agent_bit_buffer_type --  
value filesystem
```

Registry



Overview

The **Registry** component acts as a common service accessible to the other API Management - Local Edition components to discover other Local Edition components.

services, such as:

- The host/port of the service
- The properties of the Local Edition components
- The bindings of the Local Edition components

The registry also outlines the Cluster-wide topology, including:

- Cluster
- Zones
- Local Edition components
- Properties and bindings of the Local Edition components

The Registry has the following core characteristics:

- Redundancy available through the cluster
- Works with the persistence layer (for example, Cassandra) to maintain a "current snapshot" of the cluster-wide topology.
- Provides fine-grained REST APIs for each of the Local Edition components; for example, Cache, SQL DB, Traffic Manager. This includes:
 - Creation of Local Edition components
 - Listing of Local Edition components
 - Updates to Local Edition components
 - Receiving status and heartbeats from Local Edition components

The Registry is run as part of a Jetty process alongside the Cassandra process inside the NoSQL Local Edition component container. The proximity to Cassandra enables the Registry process to access Cassandra over 127.0.0.1:9042 host/port.

What's Inside

Java and Jetty Registry implementation.

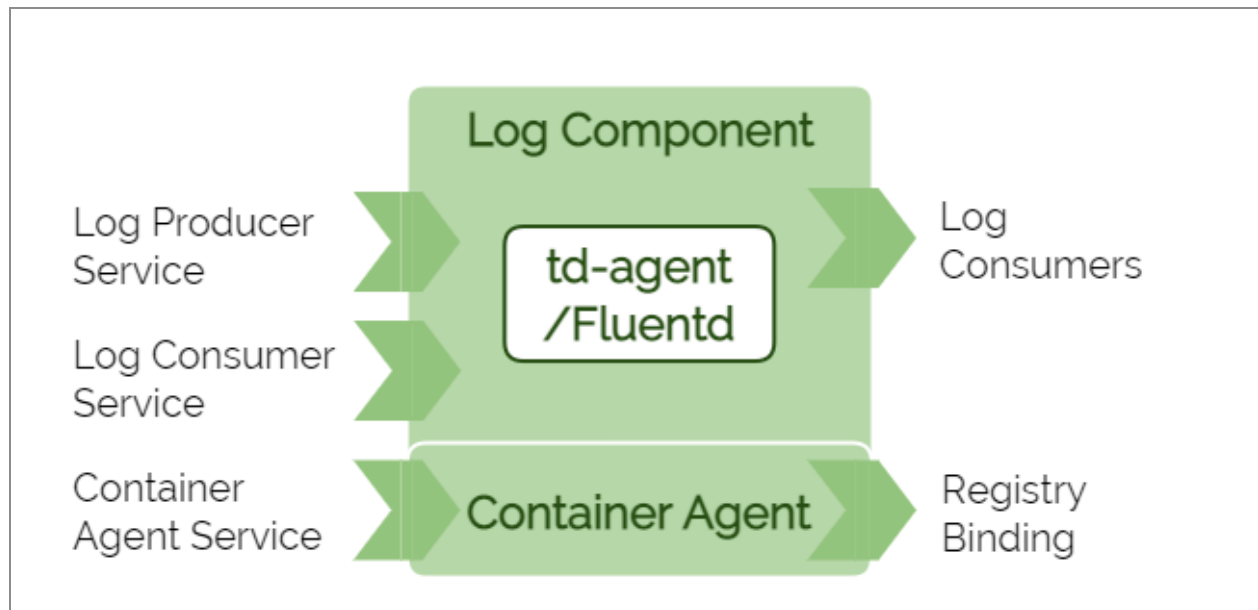
Services

Registry Service

Resource Dependencies

- NoSQL DB Binding
- Log Producer Binding

Log Service



What's Inside

- td-agent/fluentd

Services

- Log Producer Service
- Log Consumer Service

Resource Dependencies

- Log Consumers
- Registry Binding

Configuration Properties

Name	Default Value	Comments
api_key		Only needed for "Tethered" setup.
api_secret		Only needed for "Tethered" setup.
td_agent_output_channelType	DEFAULT (always enabled)	Set this property if you need to send access logs to other destinations. Accepted values are DEFAULT, FORWARD, ELASTICSEARCH, KAFKA, HTTP and TCP. Changing to anything other than DEFAULT requires additional properties: Property Name / Default Value • td_agent_out_forward_host / NULL • td_agent_out_elasticsearch_host / NULL • td_agent_out_elasticsearch_port / NULL • td_agent_out_elasticsearch_index / \${TAG} • td_agent_out_http_URI / NULL • td_agent_out_kafka_brokers / NULL • td_agent_out_kafka_topic / \${TAG} • td_agent_out_tcp_host / NULL • td_agent_out_tcp_port / NULL
mom_host	https://api-mom.mashery.com	Only needed for "Tethered" setup.
td_agent_bit_buffer_type	filesystem	The back pressure buffer type for logs - store in memory or back up to disk. In most conditions, it would not be required to change the default value; but if high disk i/o is noticed, change the setting.

Name	Default Value	Comments
		Valid values are memory and filesystem.
td_agent_bit_metrics_buffer_type	filesystem	The back pressure buffer type for metrics - store in memory or back up to disk. In most conditions, it would not be required to change the default value; but if high disk i/o is noticed, change the setting.
		Valid values are memory and filesystem.



Note: td_agent_bit_buffer_type and td_agent_bit_metrics_buffer_type properties can be set at deployment time or can be modified using the Cluster Manager CLI.

For example:

```
cm ls components
```

```
clustermanager set config --componentType <component type from ls components> --
componentId <component id from ls components> --prop td_agent_bit_buffer_type --
value filesystem
```

TD agent workers configuration

The performance of the log service is dependent on a combination of amount of traffic volume as follows; velocity of access logs, and a number of workers within the log service handling the traffic. A simple thumb rule will be - more traffic means more workers. Thus, in an expectation of amount of traffic; the log pod will dictate necessary compute, memory, and storage. The workers are distributed in sequence groups to different types of logs. An error in these values can cause the log service not to function as expected. Workers are identified by a number that starts with zero and ends with 1 index less than the number of workers. For example, when number of workers is mentioned as 10; then the workers' index starts with 0 and ends at 9. An ideal distribution of workers will ensure that access and payload(if enabled) logs get more workers than container and metrics. Each worker denotes a separate process handling a specific type of logs.

Number of workers for access logs and payloads should be same, because both work of traffic would be handling similar velocity of logs.

i Note: Each worker id must be assigned only once, we can not give the same worker id to other log types.

i Note: If the maximum worker id is n then the total number of workers would be $n+1$.

i Note: When customizing, provide workers for each log type and total number of workers.

Key	Description	Expected Values	Default value	Example
td_agent_accesslog_workers	A set of workers assigned for processing access logs. Higher the traffic, more is the number of workers required.	Range of positive value represented as a String.	5-9	td_agent_accesslog_workers="10-14"
td_agent_containerlog_workers	A set of workers assigned for processing logs of different services.	Range of positive value represented as a String.	3-4	td_agent_containerlog_workers="6-9"
td_agent_metrics_workers	A set of workers assigned for processing metrics of service running on all pods.	Range of positive value represented as a String	1-2	td_agent_metrics_workers="1-5"
td_agent_payload_logging_workers	A set of workers assigned for processing payload logs. Higher the traffic, more is the number of workers required.	Range of positive value represented as a String	10-14	td_agent_payload_logging_workers="15-19"

Key	Description	Expected Values	Default value	Example
	Note: 10-14 is assigned only for the td_agent_payload_logging_workers, it should not be used for any other workers.			
td_agent_dimdata_workers	A worker assigned for processing metadata for access logs. Recommendation: Do not assign any other worker or use 0 for any other logs.	Positive single value	0	td_agent_dimdata_workers="0"
td_agent_workers	Total number of workers to be made available	Positive integer	15	td_agent_workers="20"

Accessing QPM Data by Cluster for Untethered Deployments

QPM, or Queries per Month, is the total count of all Traffic/API calls made in a month. It includes all requests regardless of response status.

QPM Implementation

1. Every API request gets captured as an access log.
2. Each access log is transferred to the log service.
3. Every time the log service encounters an incoming access log, it increments the counter.

i Note: The counters are stored in the log pod/container at /mnt/data/trafficmanager.

4. At the turn of every 1 hour, on the zeroth minute, the counter is written to a disk.

i Note: This is similar to cron setting of 0 * * * *.

5. The counter is then reset to 0 for the counts of the next hour.

How to find the QPM

To find the total count of queries for a given month, a convenient script is available on the log pod/container at /opt/mashery/containeragent/bin/getCounts.sh.

You'll need to provide the -t| -time mm-yyyy parameter to execute the script successfully.

Usage:

```
getCounts.sh -t|--time mm-yyyy
```

Examples:

```
getCounts.sh -t 04-2019
```

or

```
getCounts.sh --time 04-2019
```

Example execution of getCounts.sh

```
/opt/mashery/containeragent/bin/getCounts.sh -t 04-2019
```

Output:

```
Counting total number of access logs
234
Finished counting of access logs
```

How to Check Compliance using QPM Implementation

1. Run the following command (for Docker Swarm) on each Log Pod/Container:

```
docker exec -it <containerid> /opt/mashery/containeragent/bin/getCounts.sh <mm-yyyy>
```

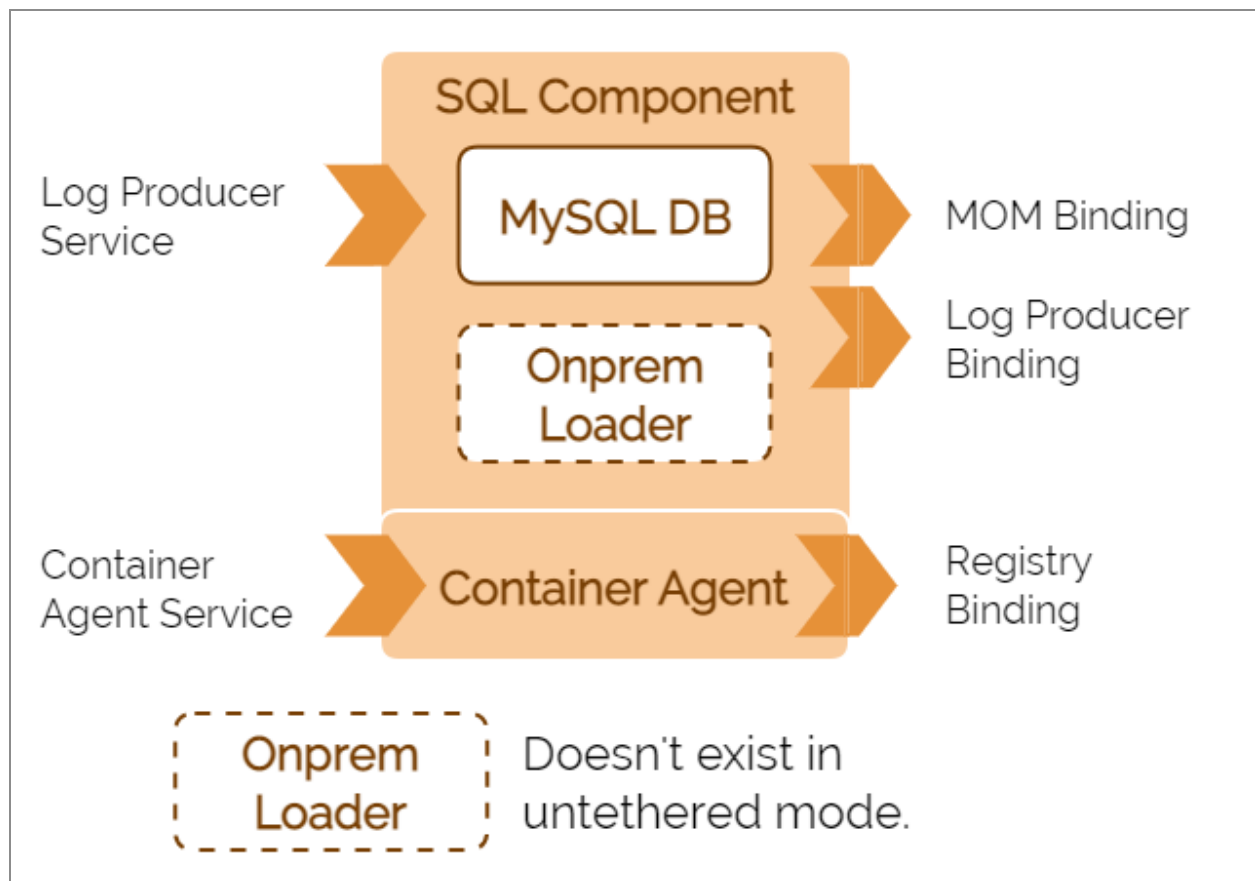
Example:

```
docker exec -it <containerid> /opt/mashery/containeragent/bin/getCounts.sh 04-2019
```

2. Add up the output for each of the pods.

$QPM = \text{LogService1\_Total} + \text{LogService2\_total} + \dots + \text{LogService}^n\_total$

SQL DB



What's Inside

- MySQL DB
- SQL DB Loader

Services

SQL DB Service

Resource Dependencies

- MOM Binding
- Log Producer Binding
- Registry Binding

Properties

Name	Default Value	Comment
api_Key		The API key needed to connect to MOM HOST in Boomi Cloud™ API Management.
api_Secret		The API password needed to connect to MOM HOST in Boomi Cloud™ API Management.
auto_binding	ON	When turned ON, logservice URLs will be auto-discovered and bound to the API Management - Local Edition component.
data_directory	/var/lib/mysql	The path to the MySQL data directory. Only the last component of the path name is created if it does not exist; the parent directory must already exist or an error occurs.

Name	Default Value	Comment
dimdataLogServiceURL	log-set-0-0.log-svc-0.default.svc.cluster.local	
expire_logs_days	30	Specifies the number of days before automatic removal of binary log files.
innodb_buffer_pool_size	128M	The size in bytes of the buffer pool, the memory area where InnoDB caches table and index data.
key_buffer_size	16M	The size of the buffer used for index blocks. The key buffer is also known as the key cache.
mysql_bin	mysql-bin	Specifies the base name to use for binary log files.
mysql_log_error	/var/log/mysqld.log	Option to determine whether the default error log destination is the console or a file, and, if a file, the file name.
mysql_log_error_verbosity	1	The verbosity for handling events intended for the error log, as filtered by the log_filter_internal error log filter component, which is enabled by default.
logserviceURL	log-set-0.log-svc.default.svc.cluster.local	Logservice URL to connect to process the logs.
max_allowed_packet	16777216	Maximum packet length to send to or receive from server.

Name	Default Value	Comment
max_binlog_size	1073741824	If a write to the binary log causes the current log file size to exceed the value of this variable, the server rotates the binary logs (closes the current file and opens the next one). The minimum value is 4096 bytes. The maximum and default value is 1GB.
mom_host	https://api-mom.mashery.com	MOM host to connect to sync data from Boomi Cloud™ API Management.
mom_connect_timeout	30000	Sets a specified timeout value, in milliseconds, to be used when opening a communications link to MOM referenced by a URLConnection.
mom_read_timeout	120000	Sets the read timeout to MOM URL to a specified timeout, in milliseconds. A non-zero value specifies the timeout when reading from Input stream when a connection is established to MOM. If the timeout expires before there is data available for read, a <code>java.net.SocketTimeoutException</code> is raised.
plugin_load	validate_password.so	This option tells the server to load the named plugins at startup. If multiple <code>--plugin-load</code> options are given, only the last one is used.

Name	Default Value	Comment
query_cache_size	8M	This system variable was removed in MySQL 8.0.3.
server_id	1234567890	Specifies the server ID. This variable is set by the <code>--server-id</code> option. The <code>server_id</code> system variable is set to 1 by default. The server can be started with this default ID, but when binary logging is enabled, an informational message is issued if you did not specify a server ID explicitly using the <code>--server-id</code> option.
service_sync_interval_minutes	15	Service sync interval to sync service configurations with MOM.
developer_sync_interval_minutes	15	Developer sync interval to sync developer configurations with MOM.
table_open_cache	1000	The number of open tables for all threads. Increasing this value increases the number of file descriptors that mysqld requires.
td_agent_bit_buffer_type	filesystem	The back pressure buffer type for logs - store in memory or back up to disk. In most conditions, it would not be required to change the default value; but if high disk i/o is noticed, change the setting. Valid values are memory and filesystem.

Name	Default Value	Comment
td_agent_bit_metrics_buffer_type	filesystem	The back pressure buffer type for metrics - store in memory or back up to disk. In most conditions, it would not be required to change the default value; but if high disk i/o is noticed, change the setting. Valid values are memory and filesystem.



Note: td_agent_bit_buffer_type and td_agent_bit_metrics_buffer_type properties can be set at deployment time or can be modified using the Cluster Manager CLI.

Example:

```
cm ls components
```

```
clustermanager set config --componentType <component type from ls components> --
componentId <component id from ls components> --prop td_agent_bit_buffer_type --
value filesystem
```

Changing a Password for MySQL Database Deployed with Docker

It is possible to change an SQL password at the time of deployment and get it deployed through the cluster. For more information, see [Configuring Properties Common to All Deployments](#).

Onprem Loader

The Onprem Loader is a program that syncs with Local Edition and loads data into the on-premises database in Local Edition. The sync process syncs the following SQL data:

- Developer classes
- Packages
- Services

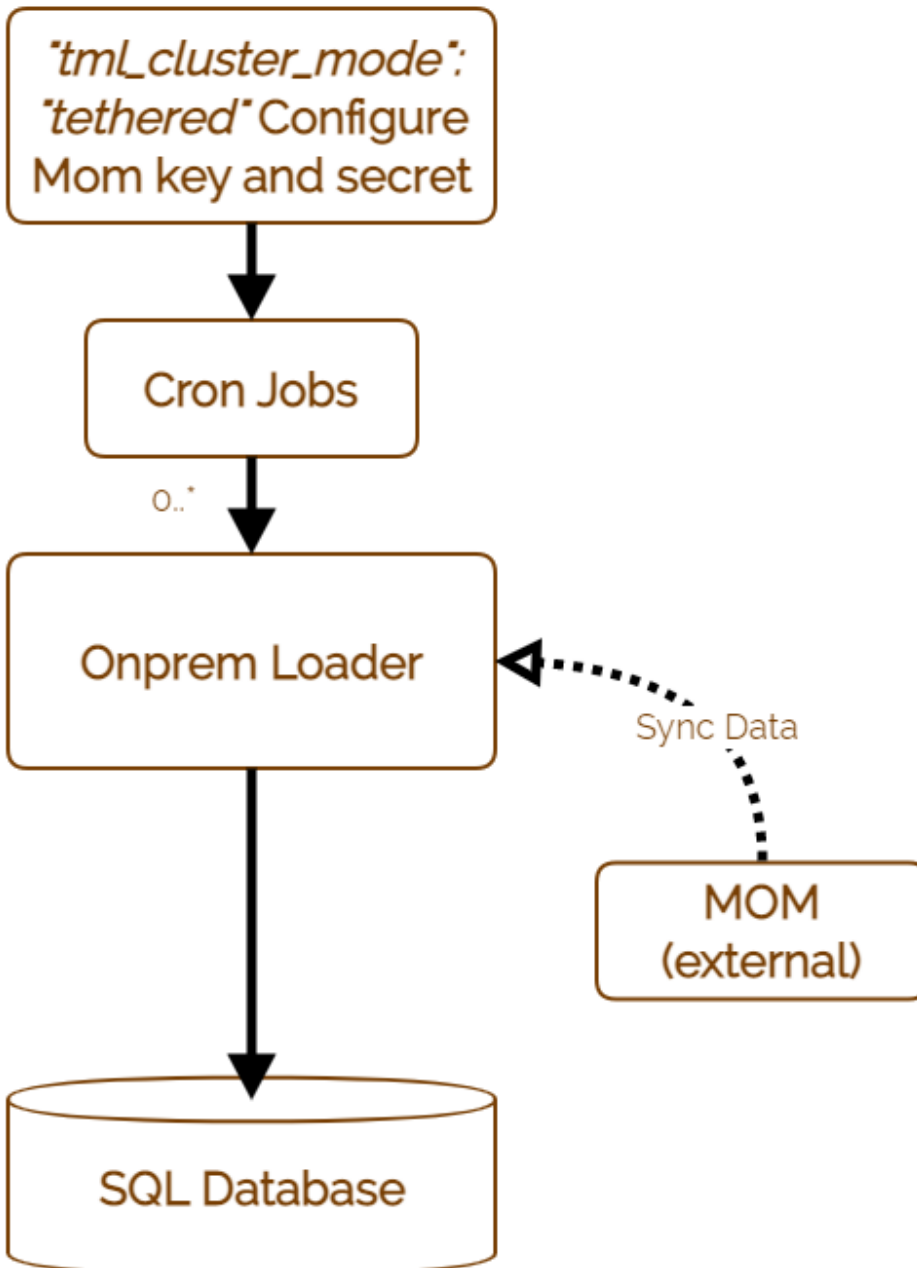
- `HttpsClientSecurity`

The following diagram describes the components for scheduling Onprem Loader jobs to storing data objects to the database.

Onprem Loader

TIBCO Cloud API

Management Local Edition



The cron jobs execute the Onprem Loader script periodically. A full sync happens every 1 hour; a partial sync happens every 15 minutes.

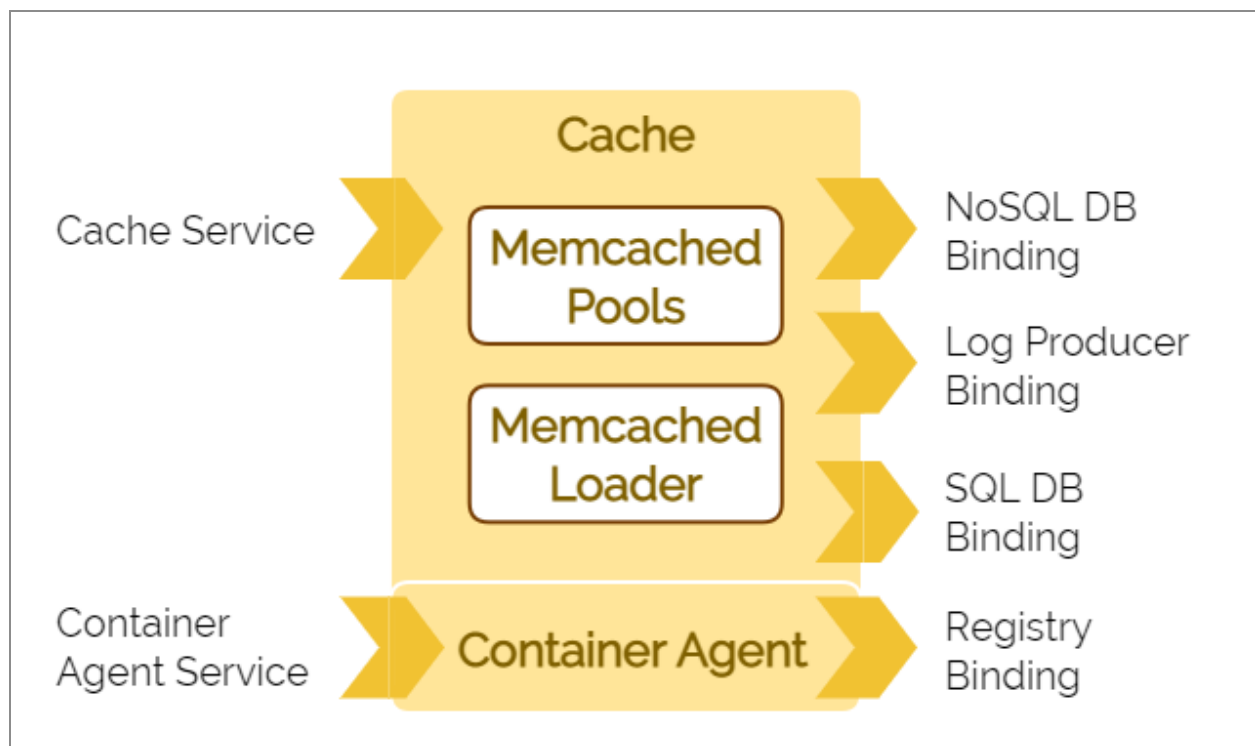
For more information on monitoring the Onprem Loader, see [Onprem Loader Monitoring](#).

Customizing the Sync Interval

To customize the sync interval for the Onprem Loader to synchronize data between Local Edition area and the MOM area, change the default values in the following properties of `tml_sql_properties.json`:

- `service_sync_interval_minutes`: 15: Service configurations to be synced with Cloud as per this interval. Value entered should be valid minutes. Default value is 15 minutes.
- `developer_sync_interval_minutes`: 15: Developer configurations to be synced with Cloud as per this interval. Value entered should be valid minutes. Default value is 15 minutes.

Cache



What's Inside

- Memcached Pools
- Memcached Loader

Services

Cache Service

Resource Dependencies

- SQL DB Binding
- NoSQL DB Binding
- Log Producer Binding
- Registry Binding

Configuration Properties

Name	Default Value	Comments
auto_binding	ON	When turned ON, SQL and logservice URLs are auto-discovered and bound to the Local Edition component.
content_cachesize	384	Cache size for memcached Content Pool
content_maxconn	10240	Max connection for memcached Content Pool
content_options	-o modern	Options for memcached Content Pool Maximum content size is 128 MB

Name	Default Value	Comments
		To set the content object size to X MB - "content_options" : "-o modern -l Xm" Note: Before changing the object size check the cache container memory allocation.
content_port	11213	Port for memcached Content Pool
content_user	nobody	User for memcached Content Pool
count_cachesize	256	Cache size for memcached Count Pool
count_maxconn	10240	Max connection for memcached Count Pool
count_options	-o modern	Options for memcached Count Pool
count_port	11212	Port for memcached Count Pool
count_user	nobody	User for memcached Count Pool memcache
logserviceURL	log-set-0.log-svc.default.svc.cluster.local	Logservice URL to connect to process the logs.
mem_cachesize	384	Cache size for

Name	Default Value	Comments
		memcached Pool
mem_maxconn	10240	Max connection for memcached Pool
mem_options	-o modern	Options for memcached Pool
mem_port	11211	Port for memcached Pool
mem_user	nobody	User for memcached Pool
packager_cachesize	128	Cache size for memcached Packager Pool
packager_maxconn	10240	Max connection for memcached Packager Pool
packager_options	-o modern	Options for memcached Packager Pool
packager_port	11215	Port for memcached Packager Pool
packager_user	nobody	User for memcached Packager Pool
services_cachesize	128	Cache size for memcached Service Pool
services_maxconn	10240	Max connection for memcached Service Pool
services_options	-o modern	Options for memcached Service Pool

Name	Default Value	Comments
services_port	11214	Port for memcached Service Pool
services_user	nobody	User for memcached Service Pool
sqlURL	mysql-set-0.mysql-svc.default.svc.cluster.local	MYSQL URL to connect to load data from
updateddevclass	10	Sync developer classes updated in the last X minutes. X is the value of this property.
updatedhttpsclientsecurity	30	Sync https client security information updated in the last X minutes. X is the value of this property.
updatedmapi	10	Sync mapis updated in the last X minutes. X is the value of this property.
updatedpackager	30	Sync packagers updated in the last X minutes. X is the value of this property.
updateservice	10	Sync services updated in the last X minutes. X is the value of this property.
td_agent_bit_buffer_type	filesystem	The back pressure buffer type for logs - store in memory or back up to disk. In most conditions, it would not be required to change the default value;

Name	Default Value	Comments
		but if high disk i/o is noticed, change the setting. Valid values are memory and filesystem.
td_agent_bit_metrics_buffer_type	filesystem	The back pressure buffer type for metrics - store in memory or back up to disk. In most conditions, it would not be required to change the default value; but if high disk i/o is noticed, change the setting. Valid values are memory and filesystem.



Note: td_agent_bit_buffer_type and td_agent_bit_metrics_buffer_type properties can be set at deployment time or can be modified using the Cluster Manager CLI.

For example:

```
cm ls components
```

```
clustermanager set config --componentType <component type from ls components> --
componentId <component id from ls components> --prop td_agent_bit_buffer_type --
value filesystem
```

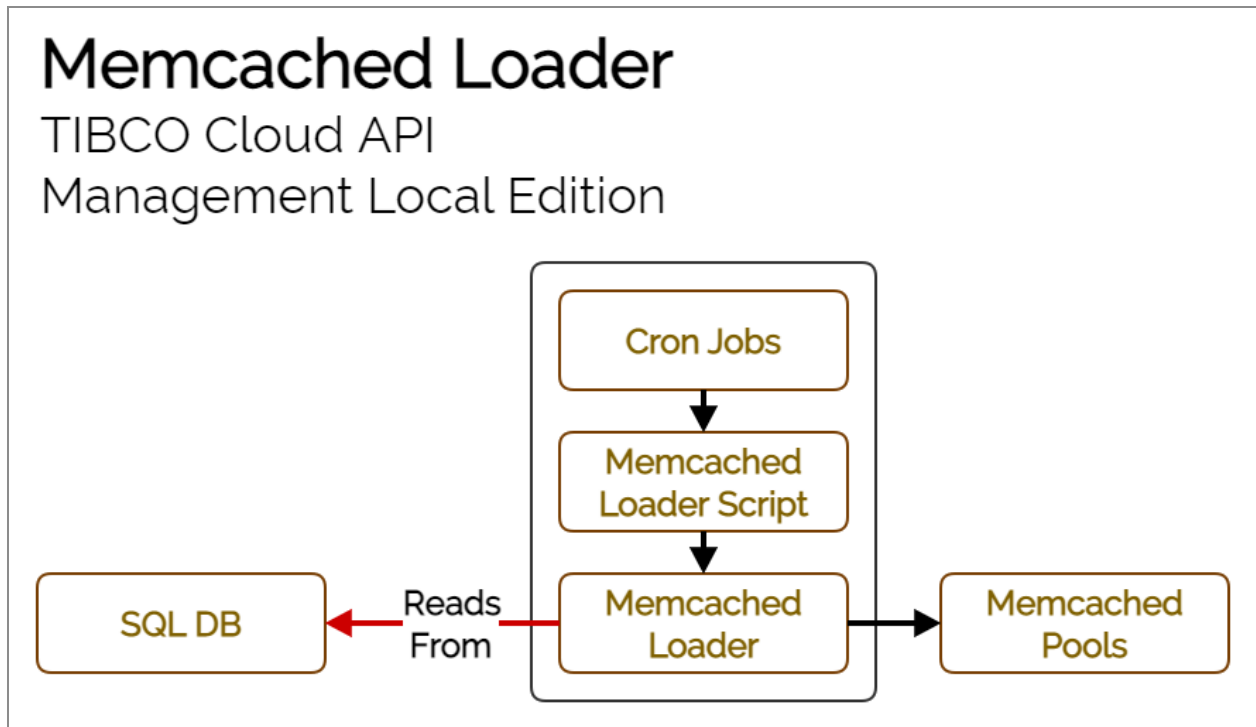
Memcached Loader

The Memcached Loader is a program that updates the following in the different Memcached pools:

- Mapis

- Developer classes
- Packages
- Services
- HttpClientSecurity

The following diagram describes the components for scheduling Memcached Loader jobs to storing data objects to memcached pools.

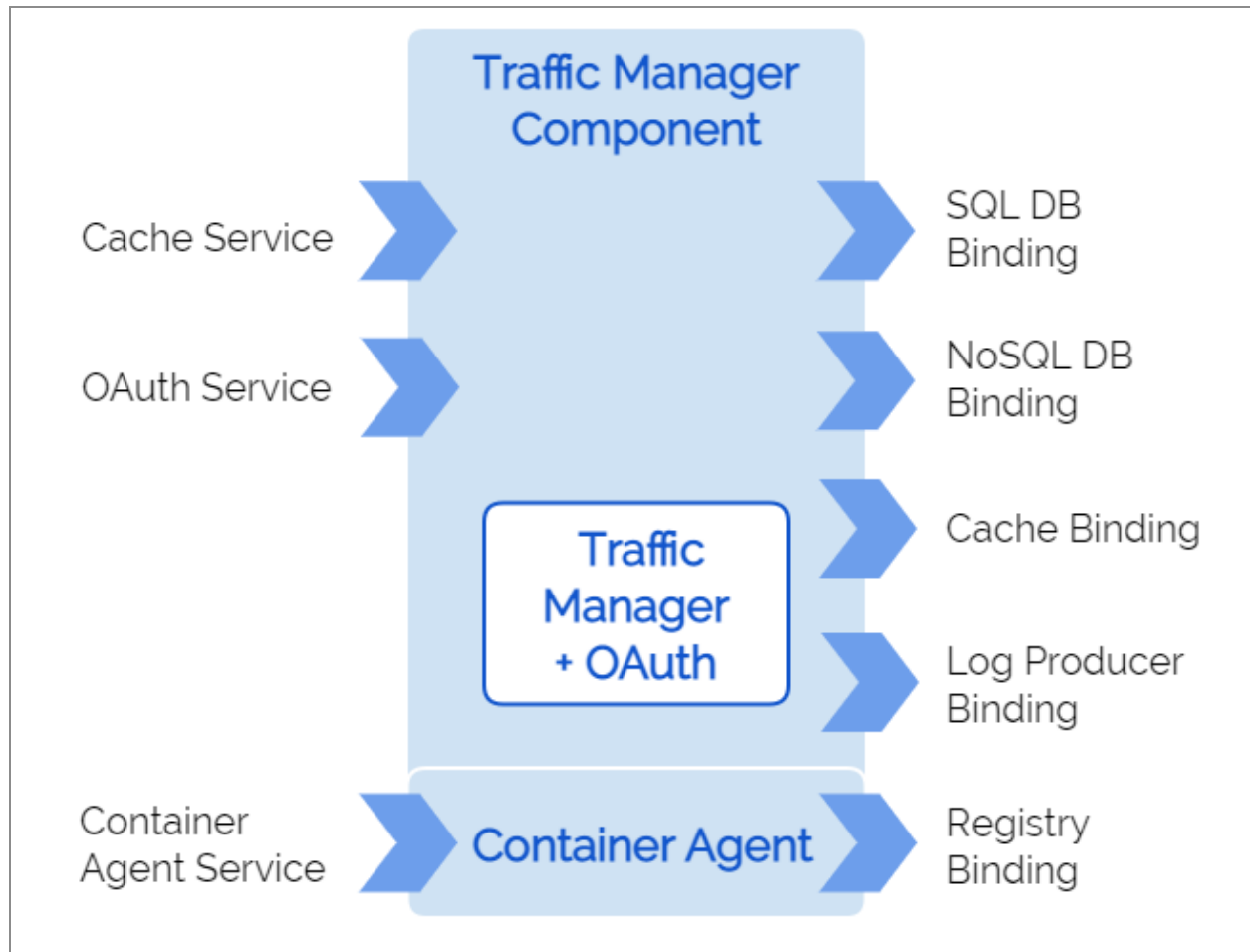


The cron jobs execute the Memcached Loader script periodically. A full sync happens every 1 hour; a partial sync happens every 15 minutes.

i Note: The timing for cron jobs is not configurable.

For more information on monitoring the Memcached Loader, see [Memcached Loader Monitoring](#).

Traffic Manager



What's Inside

- Java and Jetty Traffic Manager
- OAuth

Services

- Traffic Manager Service
- OAuth Service

Resource Dependencies

- SQL DB Binding
- NoSQL DB Binding
- Cache Binding
- Log Producer Binding
- Registry Binding

Properties

Name	Default Value	Comments
auto_binding	ON	When turned ON, sql, cache and logservice URLs will be auto-discovered and bound to the API Management - Local Edition component.
http_conn_manager_max_total	256	Caps the maximum supported backend client connections at any given time.
jetty_acceptqueuesize	10 (256 in code)	Jetty Queue size.
jetty_maxidletime	18000 (30000 in code)	Jetty Maximum Idle time.
jetty_pool_maxthreads	500 (500 in code)	Maximum number of jetty threads to be running
logserviceURL	log-set-0-0.log-svc-0.default.svc.cluster.local	Logservice URL is used to communicate with logservice component which processes the Traffic Manager logs.
mail_protocol	smtp	TCP/IP protocol used in sending and receiving e-mail.

Name	Default Value	Comments
mail_smtp_host		The SMTP server to connect to.
mail_smtp_password		Password to connect to SMTP.
mail_smtp_port	25	The SMTP server port to connect to, if the connect() method doesn't explicitly specify one.
mail_smtp_user		Default user name for SMTP.
mail_smtps_host		The SMTPS server to connect to.
mail_smtps_password		Password to connect to SMTPS.
mail_smtps_port	465	The SMTPS server port to connect to, if the connect() method doesn't explicitly specify one.
mail_smtps_user		Default user name for SMTPS.
max_uri_len_for_accesslog	2048	Limits the maximum length of URI data being written to the access log.
java_security_negative_cache_lookups	0	Specified in java.security to indicate the caching policy for unsuccessful name lookups from the name service. The value is specified as an integer to indicate the number of seconds to cache the failure for unsuccessful lookups.
java_security_cache_lookups	0	Specified in java.security to indicate the caching policy for successful name lookups from the name service. The value is specified as

Name	Default Value	Comments
		an integer to indicate the number of seconds to cache the successful lookup.
nosqlURL	cass-set-0.cass-svc.default.svc.cluster.local	NoSQL URL is used to communicate with registry service.
oauth_http_enabled	TRUE	OAuth HTTP is enabled by default.
oauth_http_port	9083	HTTP port for OAuth.
oauth_https_enabled	TRUE	OAuth HTTPS is enabled by default.
oauth_https_port	8083	HTTPS port for OAuth.
sqlURL	mysql-set-0.mysql-svc.default.svc.cluster.local	SQL URL is used to communicate with the SQL component to get the data.
vmargs	"-d64 \n-Xmx1024m"	VM arguments that typically change the behavior of the Java Virtual Machine (JVM). The Xmx argument allows the Java heap to grow.
td_agent_bit_buffer_type	filesystem	The back pressure buffer type for logs - store in memory or back up to disk. In most conditions, it would not be required to change the default value; but if high disk i/o is noticed, change the setting. Valid values are memory and filesystem.
td_agent_bit_	filesystem	The back pressure buffer type for

Name	Default Value	Comments
metrics_buffer_type		metrics - store in memory or back up to disk. In most conditions, it would not be required to change the default value; but if high disk i/o is noticed, change the setting. Valid values are memory and filesystem.
ssl_debug	false	Set this property to true, to log all SSL debug information to the runtime log at <code>/var/log/mashery/javaproxy-runtime.log</code> It is recommended to turn off the debugging once it is done, as each handshake is logged and this increases the latency.



Note: `td_agent_bit_buffer_type` and `td_agent_bit_metrics_buffer_type` properties can be set at deployment time or can be modified using the Cluster Manager CLI.

Example:

```
cm ls components
```

```
clustermanager set config --componentType <component type from ls components> --
componentId <component id from ls components> --prop td_agent_bit_buffer_type --
value filesystem
```

TLS Server Security

When Traffic Manager in Local Edition receives requests in HTTPs, Transport Layer Security (TLS) can be set in "*properties/tml_tm_properties.json*". The default settings are:

```
"includeProtocols": ["TLSv1.2", "TLSv1.3"],
"includeCipherSuites": ["TLS_DHE_RSA_WITH_AES_256_CBC_SHA", "TLS_DHE_RSA_WITH_AES_256_CBC_SHA256", "TLS_AES_128_GCM_SHA256", "TLS_AES_256_GCM_SHA384"],
"excludeProtocols": ["SSLv3", "TLSv1", "TLSv1.1"],
"excludeCipherSuites": ["TLS_ECDHE_ECDSA_WITH_RC4_128_SHA", "TLS_ECDHE_RSA_WITH_RC4_128_SHA", "SSL_RSA_WITH_RC4_128_SHA", "TLS_ECDH_ECDSA_WITH_RC4_128_SHA", "TLS_ECDH_RSA_WITH_RC4_128_SHA", "SSL_RSA_WITH_RC4_128_MD5", "TLS_ECDH_anon_WITH_RC4_128_SHA", "SSL_DH_anon_WITH_RC4_128_MD5"]
```

TLS security settings for the V3 API in Configuration Manager can be set in *"properties/tml_cm_properties.json"*. The default settings are:

```
"includeProtocols": ["TLSv1.2", "TLSv1.3"],
"includeCipherSuites": ["TLS_DHE_RSA_WITH_AES_256_CBC_SHA", "TLS_DHE_RSA_WITH_AES_256_CBC_SHA256", "TLS_AES_128_GCM_SHA256", "TLS_AES_256_GCM_SHA384"],
"excludeProtocols": ["SSLv3", "TLSv1", "TLSv1.1"],
"excludeCipherSuites": ["TLS_ECDHE_ECDSA_WITH_RC4_128_SHA", "TLS_ECDHE_RSA_WITH_RC4_128_SHA", "SSL_RSA_WITH_RC4_128_SHA", "TLS_ECDH_ECDSA_WITH_RC4_128_SHA", "TLS_ECDH_RSA_WITH_RC4_128_SHA", "SSL_RSA_WITH_RC4_128_MD5", "TLS_ECDH_anon_WITH_RC4_128_SHA", "SSL_DH_anon_WITH_RC4_128_MD5"]
```

TLS Client Security

When Local Edition communicates with target servers, the default settings of Transport Layer Security (TLS) in Traffic Manager can be customized.

Customizing Default Security Settings

The default security settings in both the affected components can be customized by following the steps below before the images are built using the Local Edition Installer:

1. Start the Local Edition Installer. For more information refer to the [Local Edition Installer](#) section.
For more information refer to the [Local Edition Installer](#) section.
2. Customize the TLS security settings for Traffic Manager:
 - a. Open the `install-core-tools.sh` file at `/var/jenkins_home/docker-build/tmhc/install/install-core-tools.sh`.

b. Make appropriate changes to the following line:

```
disabled_tls_algorithms='jdk.tls.disabledAlgorithms=SSLv3,SSLv2Hello,TLSv1,RC4,DES,DESede,MD5withRSA,DH keySize < 1024,EC keySize < 224,DES40_CBC,RC4_40'
```

c. Save the file.

3. Run the `build_docker` job to build new images.

Cluster Manager

The Cluster Manager component provides a Topology Manager CLI that accesses the registry to report on the Cluster components, their states, bindings, and properties.

Properties

Name	Default Value	Comment
td_agent_bit_buffer_type	filesystem	The back pressure buffer type for logs - store in memory or back up to disk. In most conditions, it would not be required to change the default value; but if high disk i/o is noticed, change the setting. Valid values are memory and filesystem.
td_agent_bit_metrics_buffer_type	filesystem	The back pressure buffer type for metrics - store in memory or back up to disk. In most conditions, it would not be required to change the default value; but if high disk i/o is noticed, change the setting. Valid values are memory and filesystem.

i Note: `td_agent_bit_buffer_type` and `td_agent_bit_metrics_buffer_type` properties can be set at deployment time or can be modified using the Cluster Manager CLI.

Example:

```
cm ls components
```

```
clustermanager set config --componentType <component type from ls components> --  
componentId <component id from ls components> --prop td_agent_bit_buffer_type --  
value filesystem
```

Cluster Manager CLI

Cluster Manager CLI is a command-line utility that works with the Local Edition registry API to:

- Get all Clusters:
`clustermanager list clusters`
- Use one of the clusters: `
`clustermanager use cluster <cluster-id>`
- Get all Zones:
`clustermanager list zones`
- Use one of the zones:
`clustermanager use zone <zone-id>`
- Get all nodes of different types in a zone:
`clustermanager list nodes --type=<node-type>`
- Get all nodes in a zone:
`clustermanager list nodes`

The Cluster Manager CLI is a binary that is aware of its own Cluster and Zone context.

The CLI container is also aware of how many "initial" set of components form the cluster:

- How many zones in the cluster
- How many NoSQL, SQL, Cache, Log Service, Traffic Manager (TM) components in each zone

- The Cassandra Data Centers (DCs) and the replication factor
- The registry URL

The CLI acts as the client to the following set of services:

1. The Registry APIs

The CLI uses the Registry for most of its operations such as:

- Listing of Clusters
- Listing of Zones in a cluster
- Listing of Components in a zone
- Listing of configuration properties for a Component ID
- Listing of configuration properties for all components of a Component Type in the zone, e.g. common properties across all SQL components in the zone
- Listing of configuration properties for all components of a Component Type in the cluster, e.g. common properties across all SQL components across all the zones in a cluster
- Setting configuration properties on component/zone/cluster
- Listing of Resource Bindings for a Component ID
- Setting of Resource Bindings for all components of a Component Type in the zone, e.g. bind all TMs in a zone to a specific SQL component in the zone

2. The Agent APIs

The CLI uses the Agent APIs for some of its operations that affect the lifecycle of the Agent.

- Set new configuration property - The CLI sets the Agent state to "UNSATISFIED" thereby causing the Agent to go through its lifecycle again.
- Set new resource binding - The CLI sets the Agent state to "UNSATISFIED" thereby causing the Agent to go through its lifecycle again.
- Set log level.
- Set log sync interval.
- Set verbose logging.

3. Certificate Management Local APIs

- Lists certificates/identities.

- Sets certificates/identities with new certificates.

CLI Commands

Valid Usage:

Valid Cluster Manager Usage

```
[root@cm-deploy-0-856b6c784c-x278w builder]# clustermanager
A command-line application to manage the Mashery 5.0 topology
```

Usage:

```
clustermanager [command]
```

Available Commands:

```
deploy  Deploy Mashery cluster topology
help    Help about any command
inspect Inspect the cluster deployment or configs of components in a zone
list    List of [clusters|zones|components|loggers|config|bindings|identities|certificates] in the
topology
set     Set configuration, bindings, loggers, logsyncinterval, identities & trust certificates in the
topology
use     Use available cluster or zone
version Print the version number of clustermanager CLI
```

Flags:

```
-h, --help  help for clustermanager
```

Use "clustermanager [command] --help" for more information about a command.

```
[root@cm-deploy-0-856b6c784c-x278w builder]# alias cm=clustermanager
```

Command Help:

Cluster Manager Help

```
[root@cm-deploy-0-856b6c784c-x278w builder]# cm --help
A command-line application to manage the Mashery 5.0 topology
```

Usage:

```
clustermanager [command]
```

Available Commands:

```
deploy  Deploy Mashery cluster topology
help    Help about any command
inspect Inspect the cluster deployment or configs of components in a zone
```


list List of [clusters|zones|components|loggers|config|bindings|identities|certificates] in the topology
 set Set configuration, bindings, loggers, logsyncinterval, identities & trust certificates in the topology
 use Use available cluster or zone
 version Print the version number of clustermanager CLI

Flags:

-h, --help help for clustermanager

Use "clustermanager [command] --help" for more information about a command.

Command Specific Help:

Cluster Manager Command Specific Help

[root@cm-deploy-0-856b6c784c-x278w builder]# cm deploy --help
 Deploy Mashery cluster topology using the deployment file

Usage:

clustermanager deploy [flags]

Flags:

--file string path to the cluster deployment descriptor file
 -h, --help help for deploy

[root@cm-deploy-0-856b6c784c-x278w builder]# cm inspect --help
 Inspect the cluster deployment or configs of components in a zone

Usage:

clustermanager inspect [flags]
 clustermanager inspect [command]

Available Commands:

cluster Inspect the cluster
 configs Inspect configs in a cluster

Flags:

-h, --help help for inspect

Use "clustermanager inspect [command] --help" for more information about a command.

[root@cm-deploy-0-856b6c784c-x278w builder]# cm list --help
 List of [clusters|zones|components|loggers|config|bindings|identities|certificates] in the topology

Usage:

```
clustermanager list [flags]
clustermanager list [command]
```

Aliases:

```
list, ls
```

Available Commands:

```
bindings    List of resource bindings of a specific component
certificates List of trust certificates available for the cluster & zone
clusters     List of clusters in the topology
components   List of Tibco Mashery Components in a zone
config       List of configuration settings for a specific component
identities   List of identities available for the cluster & zone
loggers      List of loggers available for a specific component
zones        List of zones in a cluster
```

Flags:

```
-h, --help  help for list
```

```
[root@cm-deploy-0-856b6c784c-x278w builder]# cm set --help
```

Set configuration, bindings, loggers, logsyncinterval, identities & trust certificates in the topology

Usage:

```
clustermanager set [flags]
clustermanager set [command]
```

Available Commands:

```
binding      Set resource binding for a component
certificate   Set certificate in the scope of a cluster/zone
config       Set configuration property in the scope of a cluster/zone/component
identity     Set identity in the scope of a cluster/zone
loglevel     Set log level in the scope of a cluster/zone/component
logsyncinterval Set log sync interval in minutes [range 1-59] in the scope of a cluster/zone
verbose      Set verbose log for traffic manager(s) in the scope of a cluster/zone/component
```

Flags:

```
-h, --help  help for set
```

```
[root@cm-deploy-0-856b6c784c-x278w builder]# cm use -h
```

Use available cluster or zone in the topology

Usage:

```
clustermanager use [flags]
clustermanager use [command]
```

Available Commands:

```
cluster  Use available cluster
zone     Use available zone in a cluster
```

Flags:

```
-h, --help  help for use
```

```
[root@cm-deploy-0-856b6c784c-x278w builder]# cm version -h
All software has versions. This is Cluster Manager CLI's
```

Usage:

```
clustermanager version [flags]
```

Version

Command: clustermanager version

Command Objective: To get the current version of the Cluster Manager

Command Usage:

Version

```
[root@a9fc970f406e bin]# clustermanager version
Mashery Cluster Manager v1.0
```

Deploy

Command: clustermanager deploy

Command Objective: To deploy a cluster topology using a deployment file

Command Help:

Deploy Help

```
[root@a9fc970f406e bin]# clustermanager deploy --help
Deploy Mashery cluster topology using the deployment file
```

Usage:

```
clustermanager deploy [flags]
```

Flags:

```
--file string  path to the cluster deployment descriptor file
-h, --help      help for deploy
```

Command Valid Usage:

Deploy Valid Usage

```
[root@a9fc970f406e bin]# clustermanager deploy
Creating the deployment..
2018/11/20 23:23:03 Deploying
{
  "clusterName": "SampleCluster",
  "zoneNames": [
    {
      "name": "local"
    }
  ],
  "tmCount": 2,
  "dsCount": 1,
  "oauthCount": 1,
  "cacheCount": 1,
  "sqlCount": 1,
  "logCount": 1,
  "nosqlCount": 1
}
Successfully created the deployment for cluster [SampleCluster]
```

Use**Command: clustermanager use**

Command Objective: To allow the user to select the scope of subsequent operations.

Valid scopes are Cluster

Command Help:

Use Help

```
[root@b6caa48479da builder]# clustermanager use --help
Use available cluster or zone in the topology
```

Usage:

```
clustermanager use [flags]
clustermanager use [command]
```

Available Commands:

```
cluster  Use available cluster
zone     Use available zone in a cluster
```

Flags:

```
-h, --help  help for use
```

Command Valid Usage:

Select Cluster Scope

```
[root@cm-deploy-0-856b6c784c-x278w builder]# cm use cluster 8d6f22d1-8797-43d6-9f54-57533ab1d560
Using cluster [Tibco Mashery Local Reference Cluster]
```

Select Zone Scope

```
[root@cm-deploy-0-856b6c784c-x278w builder]# cm use zone 535db9aa-e58c-4939-9b14-e4e5746f3b44
Using cluster [Tibco Mashery Local Reference Cluster]
Using Zone [local]
```

Command Invalid Usage:

Invalid Usage: Select Cluster Scope

```
[root@cm-deploy-0-856b6c784c-x278w builder]# cm use cluster Tibco
No matching cluster found: Tibco
```

Invalid Usage: Select Zone Scope

```
[root@cm-deploy-0-856b6c784c-x278w builder]# cm use zone local
No matching zone found: local
```

Registry APIs

List

Command: clustermanager list

Command Help:

List Help

```
[root@b6caa48479da builder]# clustermanager list --help
List of [clusters|zones|components|loggers|config|bindings|identities|certificates] in the topology
```

Usage:

```
clustermanager list [flags]
clustermanager list [command]
```

Aliases:

```
list, ls
```

Available Commands:

```
bindings  List of resource bindings of a specific component
certificates List of trust certificates available for the cluster & zone
clusters  List of clusters in the topology
components List of Tibco Mashery Components in a zone
config    List of configuration settings for a specific component
identities List of identities available for the cluster & zone
loggers   List of loggers available for a specific component
zones     List of zones in a cluster
```

Flags:

```
-h, --help  help for list
```

Subcommand: clustermanager list clusters

Subcommand Objective: To list all the available clusters.

The default list display format is "table"

Subcommand Help:

List Clusters Help

```
[root@b6caa48479da builder]# clustermanager list clusters --help
List of clusters in the topology
```

Usage:
clustermanager list clusters [flags]

Flags:
-h, --help help for clusters

Subcommand Valid Usage: List of Clusters

```
[root@cm-deploy-0-856b6c784c-x278w builder]# cm list clusters
Cluster ID          Cluster Name
-----
8d6f22d1-8797-43d6-9f54-57533ab1d560 Tibco Mashery Gat...
```

Note: The cluster name displays the leading 20 characters of the Cluster name and truncates the rest in table display.

The display format can be changed to "JSON".

List of Clusters (JSON output)

```
[root@cm-deploy-0-856b6c784c-x278w builder]# TMG_CLI_OUTPUT=json eval 'cm ls clusters'
[
  {
    "clusterId": "8d6f22d1-8797-43d6-9f54-57533ab1d560",
    "name": "Tibco Mashery Local Reference Cluster",
    "createdTime": ""
  }
]
```

Subcommand: clustermanager list zones

Subcommand Objective: To list all the available zones in the selected cluster. This command executes successfully only after selecting one of the available clusters via "use" command. (Refer to "use" command section for details.)

Subcommand Help:

List Zones Help

```
[root@b6caa48479da builder]# clustermanager list zones --help
List of zones in a cluster in the topology
```

Usage:
clustermanager list zones [flags]

Flags:
-h, --help help for zones

Subcommand Valid Usage:

List of Zones

```
[root@cm-deploy-0-856b6c784c-x278w builder]# cm ls zones
Using cluster [Tibco Mashery Local Reference Cluster]
Zone ID           Zone Name
-----
535db9aa-e58c-4939-9b14-e4e5746f3b44 local
```

Subcommand: clustermanager list components

Subcommand Objective: To list all the available components in a zone.

Subcommand Help:

List Components Help

```
[root@b6caa48479da builder]# clustermanager list components --help
List of Tibco Mashery Components in a zone a cluster in the topology
```

Usage:
clustermanager list components [flags]

Flags:

- clusterId string cluster ID of the component
- componentType string type of the component, valid types [sql, nosql, cache, trafficmanager, logservice]
- h, --help help for components
- zoneId string zone ID of the component

Subcommand Valid Usage:

List of Components in a zone


```
[root@cm-deploy-0-856b6c784c-x278w builder]# cm ls components
Using cluster [Tibco Mashery Local Reference Cluster]
Using Zone [local]
Component ID          Type      Name      Status    Last Heartbeat Received  Host
Service Port(s)
-----
5c01c72c-11a8-4bed-ba8f-2421bc931499 sql      sql      ACTIVE    Mar 12 2020 08:59:09 +0000
10.244.1.5      9080      3306
20580ccd-13b1-4662-8827-e07cf3fce567 nosql     nosql     ACTIVE    Mar 12 2020 08:59:09
+0000 10.244.1.2      9080      9042
b30b7141-29dc-4c6f-9bf6-fc0317e6f32e cache     cache     ACTIVE    Mar 12 2020 08:59:30
+0000 10.244.1.6      9080      11212,11211,11213,11214,11215,11216
a08bba64-8a39-4fa6-9fba-4d99e82816e4 trafficmanager tm        ACTIVE    Mar 12 2020 08:59:09
+0000 10.244.1.7      9080      8080
ce22559e-d6b7-4501-8575-e83fa45f91a2 logservice log        ACTIVE    Mar 12 2020 08:59:34
+0000 10.244.1.4      9080      24224
```

Subcommand: clustermanager list config

Subcommand Objective: To list component configuration properties

Subcommand Help:

List Component Properties Configuration Help

```
[root@cm-deploy-0-856b6c784c-x278w builder]# cm ls config --help
List of configuration settings for a specific component in the topology
```

Usage:

```
clustermanager list config [flags]
```

Flags:

```
--all          get all the configs from agent
--clusterId string  cluster ID of the component
--componentId string  ID of the component
--componentType string  type of the component, valid types [sql, nosql, cache, trafficmanager,
logservice]
-h, --help      help for config
--zoneId string  zone ID of the component
```

Subcommand Valid Usage:

Response prior to setting the configuration properties for a given component:

Empty Property Response

```
[root@cm-deploy-0-856b6c784c-x278w builder]# cm ls config --componentId 5c01c72c-11a8-4bed-ba8f-2421bc931499 --componentType sql
clusterId [8d6f22d1-8797-43d6-9f54-57533ab1d560] and zoneId [535db9aa-e58c-4939-9b14-e4e5746f3b44]
No settings found for the given componentId [5c01c72c-11a8-4bed-ba8f-2421bc931499] of type [sql]
```

Response after setting the configuration properties for a given component:

Valid Property List Response

```
[root@cm-deploy-0-856b6c784c-x278w builder]# cm ls config --componentId 5c01c72c-11a8-4bed-ba8f-2421bc931499 --componentType sql
clusterId [8d6f22d1-8797-43d6-9f54-57533ab1d560] and zoneId [535db9aa-e58c-4939-9b14-e4e5746f3b44]
Using cluster name [Tibco Mashery Local Reference Cluster]
Using Zone name [local]
Using Component ID [5c01c72c-11a8-4bed-ba8f-2421bc931499] of type [sql]
Property          Value          Handler          Is Encoded? File Destination
-----
DUMMY_KEY         DUMMY_VAL         false
```

List All Config Properties

```
[root@cm-deploy-0-67548f4d56-j5j45 builder]# cl ls config --componentId 27b445c3-6656-4548-aff1-30ff84d83ec9 --componentType sql --all
clusterId [3a199dba-c433-4a45-aec6-41983eccb3e8] and zoneId [a81a9fa1-834b-409d-9042-9b405c32a83e]
Agent config files are stored under /root/config-27b445c3-6656-4548-aff1-30ff84d83ec9.zip
Using cluster name [Tibco Mashery Local Example Cluster]
Using Zone name [Tibco Mashery Local Example Zone]
Using Component ID [27b445c3-6656-4548-aff1-30ff84d83ec9] of type [sql]

Property          Value          Handler          Is Encoded? File Destination
-----
apiKey            testkey123     string           false       /etc/mashery/proxy.json
apiSecret         testsecret345  string           false       /etc/proxy.json
```

i Note: If no configurations are set for a component using CLI, the registry won't have any configurations and you will see the following message.

No settings found for the given componentId <your_component_id > of type <your_component_type>

Subcommand: clustermanager list bindings

Subcommand Objective: To list Resource Bindings used by a component.

Subcommand Help:

Help for listing Resource Bindings for a Component

```
[root@cm-deploy-0-856b6c784c-x278w builder]# cm ls bindings --help
List of other components that a given component is using as its resource binding dependency
```

Usage:

```
clustermanager list bindings [flags]
```

Flags:

```
--clusterId string    cluster ID of the component
--componentId string  ID of the component
--componentType string type of the component, valid types [sql, nosql, cache, trafficmanager,
logservice]
-h, --help            help for bindings
--resourceType string type of resource binding, valid types [sql, nosql, cache, trafficmanager,
logservice]
--zoneId string       zone ID of the component
```

Subcommand Valid Usage:

List All the Resource Bindings of a Component

```
[root@cm-deploy-0-856b6c784c-x278w builder]# cm ls bindings --componentId a08bba64-8a39-4fa6-9fba-4d99e82816e4 --componentType trafficmanager --resourceType sql
clusterId [8d6f22d1-8797-43d6-9f54-57533ab1d560] and zoneId [535db9aa-e58c-4939-9b14-e4e5746f3b44]
```

Using cluster name [Tibco Mashery Local Reference Cluster]

Using Zone name [local]

Using Component ID [a08bba64-8a39-4fa6-9fba-4d99e82816e4] of type [trafficmanager]

ComponentID	Status
5c01c72c-11a8-4bed-ba8f-2421bc931499	connected

```
[root@cm-deploy-0-856b6c784c-x278w builder]# cm ls bindings --componentId a08bba64-8a39-4fa6-9fba-4d99e82816e4 --componentType trafficmanager --resourceType cache
clusterId [8d6f22d1-8797-43d6-9f54-57533ab1d560] and zoneId [535db9aa-e58c-4939-9b14-e4e5746f3b44]
Using cluster name [Tibco Mashery Local Reference Cluster]
Using Zone name [local]
Using Component ID [a08bba64-8a39-4fa6-9fba-4d99e82816e4] of type [trafficmanager]
ComponentID          Status
-----
b30b7141-29dc-4c6f-9bf6-fc0317e6f32e connected
```

Set

Command: clustermanager set

Command Objective: To update the existing component with various parameters.

Command Help:

Set Configuration Setting Help:

Set Config Help

```
[root@b6caa48479da builder]# clustermanager set config --help
Set configuration property in the scope of a cluster/zone/component

Usage:
  clustermanager set config [flags]

Flags:
  --clusterId string    cluster ID of the component
  --componentId string  ID of the component
  --componentType string type of the component, valid types [sql, nosql, cache, trafficmanager,
logservice]
  -h, --help            help for config
  --prop string         Config property name
  --value string        Config property value
  --zoneId string       zone ID of the component
```

Set Resource Binding Help:

Set Resource Binding Help

```
[root@cm-deploy-0-856b6c784c-x278w builder]# cm set binding --help
Set resource binding for a component
```

Usage:

```
clustermanager set binding [flags]
```

Flags:

```
--clusterId string    cluster ID of the component
--componentId string  ID of the component
--componentType string type of the component, valid types [sql, nosql, cache, trafficmanager,
logservice]
-h, --help            help for binding
--resourceId string   ID of the resource binding
--resourceType string type of resource binding, valid types [sql, nosql, cache, trafficmanager,
logservice]
--zoneId string       zone ID of the component
```

The following is an example:

```
[root@cm-deploy-0-856b6c784c-x278w builder]# cm set binding --componentId a08bba64-8a39-
4fa6-9fba-4d99e82816e4 --componentType trafficmanager --resourceId 5c01c72c-11a8-4bed-
ba8f-2421bc931499--resourceType sql
Updating the TML after the change
Successfully updated the TML component a08bba64-8a39-4fa6-9fba-4d99e82816e4 for
components of type trafficmanager
Successfully set the resource binding for the the component a08bba64-8a39-4fa6-9fba-
4d99e82816e4 of type trafficmanager
```

Set Verbose Logging Help:

Set Verbose Logging Help

```
[root@37268c01c6dd builder]# cm set verbose -h
Set verbose log for traffic manager(s) in the scope of a cluster/zone/component
```

Usage:

```
clustermanager set verboselog [flags]
```

Aliases:

```
verboselog, verbose
```

Flags:

```
--clusterId string    cluster ID of the component
--componentId string  ID of the component
```

<code>--enable</code>	Enable or disable verbose logging. 'enable=true' if the flag is empty (default true)
<code>-h, --help</code>	help for verbose log
<code>--interval int</code> (default -1)	Time interval to enable verbose logging. Value must be between 1-30 (minutes)
<code>--zoneId string</code>	zone ID of the component

Command Valid Usage:

Set Configuration Setting:

Set Config Property for a Component

```
[root@cm-deploy-0-856b6c784c-x278w builder]# cm set config --componentId 5c01c72c-11a8-4bed-ba8f-2421bc931499 --componentType sql --prop DUMMY_KEY --value DUMMY_VAL
Updating the TML after the change
Successfully updated the TML component 5c01c72c-11a8-4bed-ba8f-2421bc931499 for
components of type sql
Successfully set configuration property for the scope map[cluster:Tibco Mashery Local Reference
Cluster zone:local component:5c01c72c-11a8-4bed-ba8f-2421bc931499] for components of type
sql
```

Set Verbose Logging Setting:

Set Verbose Logging (Valid Usage)

```
[root@37268c01c6dd builder]# cm set verbose --interval 5
Successfully set new log level verbose logging=INFO for component map[zone:local] of type
trafficmanager
```

Command Invalid Usage:

Set Verbose logging (Invalid usage)

```
[root@37268c01c6dd builder]# cm set verbose --interval 100
Invalid verbose log interval [100], Please see the command help
```

Inspect

Command: `clustermanager inspect`

Subcommand: `clustermanager inspect cluster`

Subcommand Objective: To inspect the cluster deployment or configurations of all the components in a given zone.

Subcommand Help:

Inspect Help

```
[root@cm-deploy-0-5d8fd6fd6f-wxlqk builder]# cm inspect cluster -h
Inspect the cluster
```

```
Usage:
  clustermanager inspect cluster [flags]
```

```
Flags:
  -h, --help      help for cluster
  -p, --properties List properties for components
```

Subcommand Valid Usage:

Inspect Cluster Usage

This returns the following:

- Release version of each API Management - Local Edition component
- Bindings of each API Management - Local Edition component
- Processes running on each API Management - Local Edition component
- Optionally, the properties that each API Management - Local Edition component runs with, this is achieved by using the optional `-p` flag as shown in the example below.

```
[root@cm-deploy-0-7744b57499-2mpq6 builder]# clustermanager inspect cluster -p
{
```

```

"name": "reference-tml-cluster-single-host",
"clusterId": "b030364b-71bf-429d-be79-3de1c21fea8d",
"zones": [
  {
    "name": "us-west1-a",
    "zoneId": "4f655270-1889-48dd-83e1-bd41c3650948",
    "components": [
      {
        "type": "sql",
        "name": "mysql-set-0-0.mysql-svc-0.default.svc.cluster.local",
        "id": "8e046ab4-00a7-4355-ba7b-0c56cfe20c1c",
        "version": "5.3.0.GA",
        "host": "10.52.0.24",
        "agentPort": 9080,
        "status": "ACTIVE",
        "port": [
          3306
        ],
        "properties": null,
        "bindings": [
          {
            "tmgcId": "63ae6b86-6b37-4f21-95d6-c5678605ea4a",
            "tmgcType": "logservice",
            "status": "connected"
          }
        ],
        "processes": [
          {
            "id": "7c3fc660-080b-49fa-b45f-a716434d0faf",
            "name": "TMG-SQL",
            "type": "SQL",
            "state": "ACTIVE",
            "script": "/opt/mashery/containeragent/bin/startup_sql.sh ",
            "startTime": 1570478911737
          },
          {
            "id": "ecb6dbe0-02cb-428d-b362-e2c44291284f",
            "name": "FluentBit-LFA",
            "type": "Log",
            "state": "ACTIVE",
            "script": "/opt/mashery/containeragent/bin/startup_lfa.sh restart",
            "startTime": 1570478911777
          }
        ]
      }
    ]
  },

```



```

{
  "type": "nosql",
  "name": "cass-set-0-0.cass-svc-0.default.svc.cluster.local",
  "id": "49004100-0051-4cf0-9eaf-795a983fc0e9",
  "version": "5.3.0.GA",
  "host": "10.52.0.21",
  "agentPort": 9080,
  "status": "ACTIVE",
  "port": [
    9042
  ],
  "properties": null,
  "bindings": [
    {
      "tmgcId": "63ae6b86-6b37-4f21-95d6-c5678605ea4a",
      "tmgcType": "logservice",
      "status": "connected"
    }
  ],
  "processes": [
    {
      "id": "1097a975-508b-4f4d-90a0-51266cec7a7d",
      "name": "TMG-NoSQL",
      "type": "nosql",
      "state": "ACTIVE",
      "script": "/opt/mashery/containeragent/bin/startup_nosql.sh",
      "startTime": 1570478733913
    },
    {
      "id": "ef5563ff-1224-4c42-a309-01de8fe57270",
      "name": "FluentBit-LFA",
      "type": "Log",
      "state": "ACTIVE",
      "script": "/opt/mashery/containeragent/bin/startup_lfa.sh restart",
      "startTime": 1570478733936
    }
  ]
},
{
  "type": "cache",
  "name": "cache-set-0-0.cache-svc-0.default.svc.cluster.local",
  "id": "60e9c4b1-1eea-4c09-a0f6-4f01639a2a1e",
  "version": "5.3.0.GA",
  "host": "10.52.0.25",
  "agentPort": 9080,

```

```

    "status": "ACTIVE",
    "port": [
        11212,
        11211,
        11213,
        11214,
        11215,
        11216
    ],
    "properties": null,
    "bindings": [
        {
            "tmgcId": "8e046ab4-00a7-4355-ba7b-0c56cfe20c1c",
            "tmgcType": "sql",
            "status": "connected"
        },
        {
            "tmgcId": "63ae6b86-6b37-4f21-95d6-c5678605ea4a",
            "tmgcType": "logservice",
            "status": "connected"
        }
    ],
    "processes": [
        {
            "id": "a7bcd9a4-8909-4c2b-b870-5136b78370bf",
            "name": "TMG-Cache",
            "type": "Cache",
            "state": "ACTIVE",
            "script": "/opt/mashery/containeragent/bin/startup_cache.sh ",
            "startTime": 1570478947492
        },
        {
            "id": "e6098b0a-0889-4a7e-bd55-b9b45d83f4e7",
            "name": "FluentBit-LFA",
            "type": "Log",
            "state": "ACTIVE",
            "script": "/opt/mashery/containeragent/bin/startup_lfa.sh restart",
            "startTime": 1570478947509
        }
    ]
},
{
    "type": "trafficmanager",
    "name": "tm-deploy-0-fccdf57d7-2rmq8.tm-svc-0.default.svc.cluster.local",
    "id": "0f79b4a8-a732-459e-a044-5940f74c7d6f",

```

```

"version": "5.3.0.GA",
"host": "10.52.0.26",
"agentPort": 9080,
"status": "ACTIVE",
"port": [
    8080
],
"properties": null,
"bindings": [
    {
        "tmgcId": "8e046ab4-00a7-4355-ba7b-0c56cfe20c1c",
        "tmgcType": "sql",
        "status": "connected"
    },
    {
        "tmgcId": "49004100-0051-4cf0-9eaf-795a983fc0e9",
        "tmgcType": "nosql",
        "status": "connected"
    },
    {
        "tmgcId": "60e9c4b1-1eea-4c09-a0f6-4f01639a2a1e",
        "tmgcType": "cache",
        "status": "connected"
    },
    {
        "tmgcId": "63ae6b86-6b37-4f21-95d6-c5678605ea4a",
        "tmgcType": "logservice",
        "status": "connected"
    }
],
"processes": [
    {
        "id": "02df6bec-8151-4136-8c30-bc2b738564ce",
        "name": "TMG-TM",
        "type": "Proxy",
        "state": "ACTIVE",
        "script": "/opt/mashery/containeragent/bin/startup_proxy.sh",
        "startTime": 1570478981926
    },
    {
        "id": "2e9da9d2-a22a-40f4-8257-78852c8c5b83",
        "name": "FluentBit-LFA",
        "type": "Log",
        "state": "ACTIVE",
        "script": "/opt/mashery/containeragent/bin/startup_lfa.sh restart",
    }
]

```

```

        "startTime": 1570478981945
    }
]
},
{
    "type": "logservice",
    "name": "log-set-0-0.log-svc-0.default.svc.cluster.local",
    "id": "63ae6b86-6b37-4f21-95d6-c5678605ea4a",
    "version": "5.3.0.GA",
    "host": "10.52.0.23",
    "agentPort": 9080,
    "status": "ACTIVE",
    "port": [
        24224
    ],
    "properties": null,
    "bindings": [],
    "processes": [
        {
            "id": "037ac1e3-3d7a-4832-a259-eed04fc29d5c",
            "name": "TMG-LOG",
            "type": "log",
            "state": "ACTIVE",
            "script": "/opt/mashery/containeragent/bin/startup_logservice.sh",
            "startTime": 1570478840238
        },
        {
            "id": "fb70c0e7-2dc2-43a1-9021-f4854dfe9da5",
            "name": "FluentBit-LFA",
            "type": "Log",
            "state": "ACTIVE",
            "script": "/opt/mashery/containeragent/bin/startup_lfa.sh restart",
            "startTime": 1570478840271
        }
    ]
},
{
    "type": "configmanager",
    "name": "cm-deploy-0-6fdcf6ff69-brq9s.cm-svc-0.default.svc.cluster.local",
    "id": "cc0f6289-d28d-443a-be59-3f881ced0435",
    "version": "5.3.0.GA",
    "host": "10.52.0.22",
    "agentPort": 9080,
    "status": "ACTIVE",
    "port": [

```

```

7080
],
"properties": null,
"bindings": [
  {
    "tmgcId": "8e046ab4-00a7-4355-ba7b-0c56cfe20c1c",
    "tmgcType": "sql",
    "status": "connected"
  },
  {
    "tmgcId": "60e9c4b1-1eea-4c09-a0f6-4f01639a2a1e",
    "tmgcType": "cache",
    "status": "connected"
  },
  {
    "tmgcId": "63ae6b86-6b37-4f21-95d6-c5678605ea4a",
    "tmgcType": "logservice",
    "status": "connected"
  }
],
"processes": [
  {
    "id": "1bdbb4f4-0300-4539-8288-631f15d29956",
    "name": "TML-CM",
    "type": "API",
    "state": "ACTIVE",
    "script": "/opt/mashery/containeragent/bin/startup_api.sh",
    "startTime": 1570478800283
  },
  {
    "id": "f75a954c-2d50-47f4-87c4-41f0c4d70ede",
    "name": "TML-UI",
    "type": "UI",
    "state": "ACTIVE",
    "script": "/opt/mashery/containeragent/bin/startup_ui.sh",
    "startTime": 1570478800303
  },
  {
    "id": "8a47487b-cf8d-4309-9f93-d302683fa92e",
    "name": "FluentBit-LFA",
    "type": "Log",
    "state": "ACTIVE",
    "script": "/opt/mashery/containeragent/bin/startup_lfa.sh restart",
    "startTime": 1570478800369
  }
]

```

```

    }
  }
}

```

Subcommand: clustermanager inspect configs

Subcommand Objective: To inspect the configurations of all components in a given zone.

If the `zone_name` parameter is not specified in the command, the current deployed zone is used. It collects the default and runtime configurations from each API Management - Local Edition agent and respective configurations from registry. It compares default and runtime configurations, and if there is any change, it compares the modified configuration with the registry. If the configuration value in runtime is not the same as the registry (values might be set through CLI and not null), the configuration is invalid, otherwise the configuration is valid.



Note: If a component was not in an ACTIVE state at least once during its lifecycle, there won't be runtime configurations to inspect. So, validation of the configurations won't be possible on that component.

Subcommand Valid Usage:

Inspect Configs for a Zone Output

```

[root@b6caa48479da builder]# clustermanager inspect configs
{
  "name": "Sample Cluster",
  "clusterId": "86db18a2-b28a-4724-96ca-faef6f5a078b",
  "zones": [
    {
      "name": "us-east-1e",
      "zoneId": "86665b5d-0919-4032-8ce0-805fd7e0e15f",
      "components": [
        {
          "type": "nosql",
          "name": "nosql",
          "id": "b6cd77bf-92df-4633-898e-a365af329ade",
          "host": "100.96.2.5",

```

```

    "validatedProperties": [
      {
        "property": "logserviceURL",
        "isValid": true,
        "registryValue": null,
        "runtimeValue": "100.96.3.11",
        "defaultValue": "log-set-0-0.log-svc-0.default.svc.cluster.local"
      }
    ]
  },
  {
    "type": "cache",
    "name": "cache",
    "id": "d0358516-57f8-4d50-b684-2c0391fabe03",
    "host": "100.96.2.4",
    "validatedProperties": [
      {
        "property": "logserviceURL",
        "isValid": true,
        "registryValue": null,
        "runtimeValue": "100.96.3.11",
        "defaultValue": "log-set-0-0.log-svc-0.default.svc.cluster.local"
      },
      {
        "property": "sqlURL",
        "isValid": true,
        "registryValue": null,
        "runtimeValue": "100.96.1.5",
        "defaultValue": "mysql-set-0-0.mysql-svc-0.default.svc.cluster.local"
      }
    ]
  },
  {
    "type": "sql",
    "name": "sql",
    "id": "f2c92ee6-d083-471d-8bf6-950266f21ceb",
    "host": "100.96.1.5",
    "validatedProperties": [
      {
        "property": "logserviceURL",
        "isValid": true,
        "registryValue": null,
        "runtimeValue": "100.96.3.11",
        "defaultValue": "log-set-0-0.log-svc-0.default.svc.cluster.local"
      },
      {

```

```

        "property": "apiSecret",
        "isValid": true,
        "registryValue": null,
        "runtimeValue": "8UN8j8BN6f",
        "defaultValue": ""
    },
    {
        "property": "apiKey",
        "isValid": false,
        "registryValue": "123KJGHGG",
        "runtimeValue": "8UJBAUIUF",
        "defaultValue": ""
    }
]
}
]
}
]
}
}

```

Certificate Management Local APIs

List

Command: `clustermanager list`

Subcommand: `clustermanager list certificates`

Subcommand Description: To list all the certificates in the selected zone within the cluster.

Subcommand Help:

List all Certificates Help

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

Subcommand Valid Usage:

List All Certificates

```
[root@a9fc970f406e bin]# clustermanager ls certificates
```

UUID	Name	Area ID	Expiration
32330473-69e5-4f93-a4c5-104601c416d7	acme.client.example.com	0	2116-05-28

Subcommand: clustermanager list identities

Subcommand Objective: To list all the identities in the selected zone within the cluster.

Subcommand Help:

List all Identities Help

```
[root@aa5de66f2645 ~]# clustermanager ls identities --help
```

List of identities available for the cluster & zone

Usage:

clustermanager list identities [flags]

Flags:

-h, --help help for identities
 --zoneId string zone ID of the component

Subcommand Valid Usage:

List All Identities

```
[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
```

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

Set

Command: clustermanager set

Subcommand: clustermanager set certificate

Subcommand Objective: To update a certificate in the selected zone within the cluster.

Subcommand Help:

Update Certificate Help

```
[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

The following is an example:

```
[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
```

Subcommand Valid Usage:

Create New Certificate

```
[root@aa5de66f2645 ~]# clustermanager create certificate --file /usr/local/bin/sample-cert.cer
```

Update Existing Certificate

```
[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
```

The following is an example:

```
[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
```

Subcommand Invalid Usage:

Invalid Usage: file not set

```
[root@aa5de66f2645 ~]# clustermanager set certificate
Error: required flag(s) "file" not set
Usage:
  clustermanager set certificate [flags]

Aliases:
```

certificate, cert

Flags:

- clusterId string cluster ID of the component
- file string certificate file to upload, CER type
- h, --help help for certificate
- uuid string unique ID for the existing certificate that needs update
- zoneId string zone ID of the component

required flag(s) "file" not set

Invalid Usage: certificate file not present

```
[root@aa5de66f2645 ~]# clustermanager set certificate --file sample.cer
No such certificate sample.cer
```

Subcommand: clustermanager set identity

Subcommand Objective: To update an identity in the selected zone within the cluster.

Subcommand Help:

Update Identity Help

```
[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

The following is an example:

```
[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
```

Subcommand Valid Usage:

Create New Identity

```
[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
```

The following is an example:

```
[root@01294de80b3d builder]# clustermanager create identity --file sample.p12 --password
password
Creating an identity for the topology
Successfully created identity for the topology
```

Update an Existing Identity

```
[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

The following is an example:

```
[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
```

Setting verifyHostname values

Flag: true

```
{code:java}
```

```
[root@2f3252898633 builder]# cm set identity --file sample.p12 --password password --uuid
cb4d2bc2-f863-440b-0020-1d11369404d8 --verifyHostname true
```

Updating identity for cluster ID 6a8bf4d8-810f-417b-8900-cb3acf2caff0 and zone ID e2844a28-2d61-4dcc-97d8-6f56a641b7a6

Successfully updated identity for cluster ID 6a8bf4d8-810f-417b-8900-cb3acf2caff0 of zone ID e2844a28-2d61-4dcc-97d8-6f56a641b7a6

```
{code}
```

Flag: false

```
{code:java}
```

```
[root@2f3252898633 builder]# cm set identity --file sample.p12 --password password --uuid
cb4d2bc2-f863-440b-0020-1d11369404d8 --verifyHostname=false
```

Updating identity for cluster ID 6a8bf4d8-810f-417b-8900-cb3acf2caff0 and zone ID e2844a28-2d61-4dcc-97d8-6f56a641b7a6

Successfully updated identity for cluster ID 6a8bf4d8-810f-417b-8900-cb3acf2caff0 of zone ID e2844a28-2d61-4dcc-97d8-6f56a641b7a6

```
{code}
```

Subcommand Invalid Usage:

Invalid Usage: file & password not set

```
[root@aa5de66f2645 ~]# clustermanager set identity
Error: required flag(s) "file", "password" not set
Usage:
  clustermanager set identity [flags]

Aliases:
  identity, idty

Flags:
  --clusterId string  cluster ID of the component
  --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
  --zoneId string     zone ID of the component

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

Invalid usage: password not set

```
[root@aa5de66f2645 ~]# clustermanager set identity --file sample.p12
Error: required flag(s) "password" not set
Usage:
  clustermanager set identity [flags]

Aliases:
  identity, idty

Flags:
  --clusterId string  cluster ID of the component
  --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
  --zoneId string     zone ID of the component

required flag(s) "password" not set
```

Invalid Usage: identity file not present

```
[root@aa5de66f2645 ~]# clustermanager set identity --file sample.p12 --password 1.tibco
No such identity sample.p12
```

CLI Help

```
[root@cm-deploy-0-5c898c8d5f-wx78f builder]# clustermanager -h
A command-line application to manage the Mashery 5.0 topology
```

Usage:

```
clustermanager [command]
```

Available Commands:

```
cluster      Execute commands against the deployment in a cluster zone
create       Create identity or certificate in available cluster or zone
deploy       Deploy Mashery cluster topology
help         Help about any command
import       Import properties configuration in the topology
inspect      Inspect the cluster deployment or configs of components in a zone
list         List of [clusters|zones|components|loggers|config|bindings|identities|certificates] in
the topology
set          Set configuration, bindings, loggers, logsyncinterval, identities & trust certificates
in the topology
stage-portal-customizations Decompress and upload portal customizations from given file location
use          Use available cluster or zone
version      Print the version number of clustermanager CLI
```

Flags:

```
-h, --help  help for clustermanager
```

Use "clustermanager [command] --help" for more information about a command.

CLI Command for Cluster Status

```
[root@cm-deploy-0-5c898c8d5f-wx78f builder]# clustermanager cluster status
*** Checking the status of Cluster 'reference-tml-cluster-single-host' ***
*** Checking the status of zone 'us-west1-a' ***
TML component [a3d8c76b-7b01-401a-b2d8-32363a3af913] of type [nosql] found with status
[ACTIVE] on Host IP [10.60.0.23]. See details below
Component ID      Component Name  Component IP  Component Status  Pending
Activity  Error
```



```
-----
a3d8c76b-7b01-401a-b2d8-32363a3af913 nosql      10.60.0.23      Not Ready
```

Sample for the cluster status command as the cluster gets healthy:

```
[root@cm-deploy-0-5c898c8d5f-wx78f builder]# clustermanager cluster status
*** Checking the status of Cluster 'reference-tml-cluster-single-host' ***
*** Checking the status of zone 'us-west1-a' ***
TML component [a3d8c76b-7b01-401a-b2d8-32363a3af913] of type [nosql] found with status
[ACTIVE] on Host IP [10.60.0.23]..
TML component [143d47f5-ee40-4985-abd5-b60c624e8986] of type [logservice] found with status
[ACTIVE] on Host IP [10.60.0.25]..
TML component [f589716e-5a2f-4ace-916b-f2febf42f7d5] of type [sql] found with status [ACTIVE]
on Host IP [10.60.0.26]..
TML component [720222d7-9603-4a4f-9c9c-d32a47e4748e] of type [cache] found with status
[ACTIVE] on Host IP [10.60.0.27]..
TML component [e068e07f-dcaa-4528-b226-7ad3b8a06565] of type [trafficmanager] found with
status [RESOLVED] on Host IP [10.60.0.28]. See details below
Component ID          Component Name    Component IP    Component Status  Pending
Activity   Error
-----
e068e07f-dcaa-4528-b226-7ad3b8a06565 tm          10.60.0.28      Not Ready
```

When the Cluster is healthy overall, the following status is seen:

```
[root@cm-deploy-0-5c898c8d5f-wx78f builder]# clustermanager cluster status
*** Checking the status of Cluster 'reference-tml-cluster-single-host' ***
*** Checking the status of zone 'us-west1-a' ***
TML component [a3d8c76b-7b01-401a-b2d8-32363a3af913] of type [nosql] found with status
[ACTIVE] on Host IP [10.60.0.23]..
TML component [143d47f5-ee40-4985-abd5-b60c624e8986] of type [logservice] found with status
[ACTIVE] on Host IP [10.60.0.25]..
TML component [f589716e-5a2f-4ace-916b-f2febf42f7d5] of type [sql] found with status [ACTIVE]
on Host IP [10.60.0.26]..
TML component [720222d7-9603-4a4f-9c9c-d32a47e4748e] of type [cache] found with status
[ACTIVE] on Host IP [10.60.0.27]..
TML component [e068e07f-dcaa-4528-b226-7ad3b8a06565] of type [trafficmanager] found with
status [ACTIVE] on Host IP [10.60.0.28]..
TML component [d8c24dd6-6f38-4474-a30c-1f2b33179340] of type [configmanager] found with
status [ACTIVE] on Host IP [10.60.0.24]..

Cluster 'reference-tml-cluster-single-host' is ready
```

CLI Command to remove unreachable data from old nodes

Command name : cleanupUnreachable

When the cluster is removed, the following status is seen:

```
clustermanager cluster cleanupUnreachable --confirm
*** Deleting the status of Cluster 'reference-tml-cluster-single-host' ***
*** Cleaning up unreachable components in zone 'us-central1-a' ***
Successfully deleted unreachable components in cluster [afa81ab5-4ff6-4cb8-b52c-c4698b2ed9cf]
and zone [8361d04c-73e1-4282-a8aa-d44cff1b02db]

All unreachable components cleaned up
[root@cm-deploy-0-5dd578987d-lqqc2 builder]# cl ls components -u
Using cluster [reference-tml-cluster-single-host]
Using Zone [us-central1-a]
Component ID           Component Type   Component Name   Component Status
Component Host   Component Agent Port   Component Service Port(s)
-----
[root@cm-deploy-0-5dd578987d-lqqc2 builder]#
```

Import of Configuration

The following command allows importing of configuration and applies it to components of different types, for example, to import configuration for a new log destination, the following needs to be done:

1. Prepare the new log destination configuration in following JSON format:

```
{
  "td_agent_output_channelType": "DEFAULT",
  "td_agent_container_output_channelType": "DEFAULT",
  "td_agent_metric_output_channelType": "DEFAULT,KAFKA",
  "td_agent_verbose_output_channelType": "DEFAULT",
  "td_agent_out_kafka_brokers": "10.97.242.197:9092",
  "td_agent_out_kafka_topic": "ml5_access_logs",
  "td_agent_out_metric_kafka_brokers": "10.97.242.197:9092",
  "td_agent_out_metric_kafka_topic": "ml511_metric_Kafka"
}
```

Let us call this file `/var/config/props.json`, as an example.

2. Use the `clustermanager import` command to import the same for the logservice container(s).

```
clustermanager import config --componentType logservice --file
/var/config/props.json
```

i Note: The JSON to be imported can only have a first-level simple key:value JSON object. It does not allow complex nested JSON objects or array of objects.

Importing help

```
[root@cm-deploy-0-5c898c8d5f-wx78f builder]# clustermanager import -h
Import properties configuration in the topology
```

Usage:

```
clustermanager import [flags]
clustermanager import [command]
```

Available Commands:

```
config    Import properties configuration in the scope of a cluster/zone/component
```

Flags:

```
-h, --help  help for import
```

```
[root@cm-deploy-0-5c898c8d5f-wx78f builder]# clustermanager import config -h
Import properties configuration in the scope of a cluster/zone/component
```

Usage:

```
clustermanager import config [flags]
```

Flags:

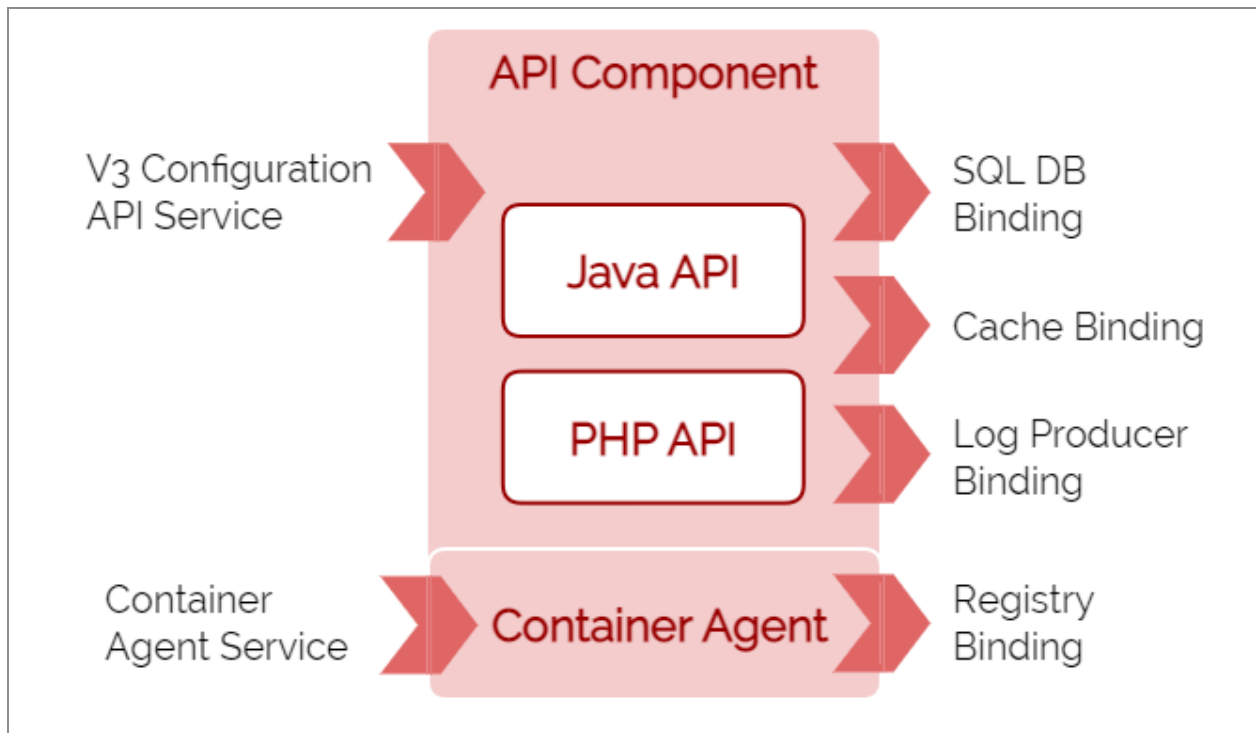
```
--clusterId string    cluster ID of the component
--componentId string  ID of the component
--componentType string type of the component, valid types [sql, nosql, cache, trafficmanager,
logservice, platformapi, configmanager]
--file string         Path to properties key-value JSON file
-h, --help            help for config
--zoneId string       zone ID of the component
```

Cluster Manager Command

The "clustermanager cluster status" command reports status of all components across all the zones of deployment that belong to the Local Edition cluster. If the status of a

component is not ACTIVE, the status command reports the data that points to any pending activity that the component is waiting to finish before becoming ACTIVE. Once all the components are ACTIVE, then the command reports that the cluster is ready.

API Management - Local Edition API



The Local Edition API component works with the Local Edition Registry using the Agent.

What's Inside

- Java API
- PHP API

Services

V3 Configuration API service

Resource Dependencies

- SQL DB Binding
- Cache Binding
- Log Producer Binding
- Registry Binding

Configuration Properties

Name	Default Value	Comment
auto-binding	ON	When turned ON, logservice URLs will be auto-discovered and bound to the Local Edition cluster component.
domain_notification_from	noreply@mashery.com	
domain_notification_recipient	papi@mashery.com	
http_conn_manager_max_total	1000	Sets the maximum supported backend client connections at any given time.
jetty_acceptqueuesize	256	Jetty Queue size
jetty_pool_maxthreads	500	Maximum number of Jetty threads to be running
logserviceURL	log-set-0-0.log-svc-0.default.svc.cluster.local	Logservice URL to connect to for processing the logs.
max_endpoints_per_service_created	600	Maximum endpoints in a service that can be created.

Name	Default Value	Comment
max_services_created	600	Maximum services that can be created.
max_uri_len_for_accesslog	2048	This property is used to limit the maximum length of uri data being written to the access log.

Configuration Manager Overview

Local Edition Configuration Manager is a browser-based configuration utility that allows administrators to easily import and manage new or existing configurations for Local Edition, with no reliance on Boomi Cloud™ API Management. Once installed and launched from a web browser, Configuration Manager provides un-tethered operation. This allows the site administrator to activate new Local Edition zones. An administrator is able to easily create and update Service and Package definitions, as well as set up and manage key issuance.

For more information on installing and using the utility, see [Configuration Manager](#).

Developer Portal Overview

Deliver a secure, on-premises Developer Portal deployed on Local Edition's API management platform. You can use the out-of-the-box configurations (pre-defined Local Edition templates) to give your developers self-service portal access, or customize a Developer Portal yourself for a uniquely-branded experience.

Similar to the Boomi Cloud™ API Management Developer Portal, Local Edition's Developer Portal features include the ability to create interactive documentation and manage API keys.

For more information on installing and using this feature, see [Developer Portal](#).

Local Edition Installer

Installer Overview

The Local Edition Installer is designed to be used to perform the following:

1. **Prebuild**: upload customizations to be built into Local Edition Docker images;
2. **Build and Verify**: build Local Edition Docker images and verify the images before proceeding to the next step;
3. **Prepush**: configure access to supported platforms before the Local Edition Docker image push and Local Edition cluster deployment;
4. **Push**: push Local Edition Docker images to supported platforms;
5. **Deploy**: customize Local Edition cluster deployment.

HTTP Proxy Support

The Local Edition installer can be run from behind an HTTP proxy server.

The example provided below uses the following addresses:

- HTTP proxy host: 10.250.1.40
- Local Edition installer Docker host: 10.250.1.42

To run the Local Edition installer from behind an HTTP proxy, an Administrator must set up the HTTP proxy in the Docker host, as well as in the Local Edition installer, as described below.

Setting up Proxy support for tml-installer

To enable proxy support for tml-installer, follow the below steps:

1. Update the proxy settings in the docker file
 - a. vi `~/docker/config.json`
 - b. To update the sample proxy, see <https://docs.docker.com/network/proxy/>

```
{
  "proxies":
  {
    "default":
    {
      "httpProxy": "http://192.168.1.12:3128",
      "httpsProxy": "http://192.168.1.12:3128",
      "noProxy": "/*.test.example.com,.example2.com,127.0.0.0/8"
    }
  }
}
```

2. In case, the firewall is running on Docker Host Machine, add a firewall rule to add the required ports

```
firewall-cmd --zone=public --add-port=80/tcp
firewall-cmd --zone=public --add-port=443/tcp
firewall-cmd --reload
```

3. Start Installer

Set up an HTTP Proxy in the Local Edition Installer

After the Local Edition installer distribution package is extracted, set up the HTTP proxy settings in the `tml-installer.env` file. The following is an example:

```
TERM=xterm
# HOST_IP is the IP of docker host, loopback address does not work
HOST_IP=10.250.1.42

# http proxy, i.e. http://10.250.1.40:3128
http_proxy=http://10.250.1.40:3128
# https proxy, it is usually the same as HTTP_PROXY
https_proxy=http://10.250.1.40:3128
# no proxy, i.e. 127.0.0.1, localhost
no_proxy=127.0.0.1, localhost
```


HTTP Proxy Support with Authentication

The Local Edition installer can be run from behind an HTTP proxy server with authentication.

The example uses the following sample information:

Parameter	Value
HTTP proxy host	10.250.1.40
HTTP proxy user name	squiduser
HTTP proxy password	squidpwd
Local Edition installer Docker host	10.250.1.42

To run the Local Edition installer from behind an HTTP proxy with authentication, an administrator must set up the HTTP proxy in the Docker host, as well as in the Local Edition installer, as follows:

Set up an HTTP Proxy in the Docker Host

This involves setting up the HTTP proxy at the operating system level, as well as in the Docker service.

- **HTTP proxy settings on operating system level** - On Linux, set up the HTTP proxy in the `/etc/environment` file. The following is an example:

```
http_proxy="http://squiduser:squidpwd@10.250.1.40:3128"  
https_proxy="http://squiduser:squidpwd@10.250.1.40:3128"  
no_proxy="127.0.0.1, localhost"
```

- **HTTP proxy settings for the Docker service** - This varies from one Linux distribution to another. The following are HTTP proxy settings for a Docker Service on CentOS 7:
 1. Create folder `/etc/systemd/system/docker.service.d`.
 2. Create an `http-proxy.conf` file in the folder created in step 1 and add the following content:

```
[Service]
Environment="HTTP_PROXY=http://squiduser:squidpwd@10.250.1.40:3128/" "NO_PROXY=localhost,127.0.0.1,10.250.1.*"
```

3. Restart the Docker service.

Set up an HTTP Proxy in the Local Edition Installer

After the Local Edition installer distribution package is extracted, set up the HTTP proxy settings in the `tml-installer.env` file. The following is an example:

```
TERM=xterm
# HOST_IP is the IP of docker host, loopback address does not work
HOST_IP=10.250.1.42

# http proxy, i.e. http://10.250.1.40:3128
http_proxy=http://squiduser:squidpwd@10.250.1.40:3128

# https proxy, it is usually the same as HTTP_PROXY
https_proxy=http://squiduser:squidpwd@10.250.1.40:3128

# no proxy, i.e. 127.0.0.1, localhost
no_proxy=127.0.0.1, localhost
```

Once you set up an HTTP proxy in the installer, you must start the installer. To completely add an HTTP Proxy Authentication support, follow the steps specified in [Setting up HTTP Proxy Authentication in Installer Jenkins](#).

Deploying the Installer

Prerequisites

Local Edition Installer runs as a Docker container on the host. The prerequisites to deploy the Installer are:

- Docker host should run Linux or MacOS.
- Docker host should have Docker 18.09.5 (or later) installed and running.
- Docker host must have port 8080 open in firewall to connect to Jenkins web console.

- Docker host must have enough storage (recommend 200GB) for Docker container's volume.

i Note: The Installer should not be deployed on the same virtual machine that is expected to run a Local Edition 5.x deployment.

i Note: There is an option to [run the installer on Windows](#).

Deploying the Installer

An Admin can deploy and manage the installer as root in the Docker host.

To deploy and manage the Installer as a non-root user in the Docker host, use the following procedure:

1. Create a non-root user:

```
# create user "tmlops"
useradd tmlops#
change user "tmlops" password
passwd tmlops
```

2. Add the non-root user to the Docker group:

```
# add user "tmlops" to group "docker"
usermod -a -G docker tmlops

# verify the groups of user "tmlops"
groups tmlops

# example output
# tmlops : tmlops docker
```

3. Get the Installer distribution package and extract it to the host. For example: tar xvf TIB_mash-local_5.2.0.111_cf.tar.gz

How to Run the Installer on Windows

The following section describes how to set up and launch the API Management - Local Edition Installer on Windows.

Before you begin

Docker desktop for Windows installed and started correctly.

i Note: Follow the instructions at <https://docs.docker.com/docker-for-windows/install/>.

Setup

1. Download TIB_mash-local_5.x.x_cf.tar.gz.
2. Unzip the downloaded file in Windows to TIB_mash-local_5.x.x_cf.tar.
3. Unzip again TIB_mash-local_5.x.x_cf.tar to TIB_mash-local_5.0.0.101_cf.
4. Place start-installer.bat inside the directory TIB_mash-local_5.0.0.101_cf that is at the same location as the start-installer.sh file.

Steps to Launch Installer

Procedure

1. Open a command prompt and go the location of batch file to start the installer.
2. Enter the command in command prompt >start-installer.bat <IP Address>
 Installer displays the following prompt: You must read and accept the License agreement before proceeding.Press **Enter** for more and **q** to finish reading.
3. Read the license file? [Y,N]?"
 Select "N" for No to exit the installer. Select "Y" for Yes, to have the installer load the license file.
4. Press **Enter** to read the content of license and "q" to quit reading the license content.
5. Once the license has been read or you've quit reading the license, the installer displays the following prompt: Do you accept the license? [Y,N]?"
 Select "N" for "No to exit the installer. Select "Y" for "Yes", to have the installer will place a "license_accepted" file in the same directory with the content confirming that you've accepted the license on the particular day.
6. The installer will then start loading the image and installation starts.

7. You can choose to exit the installer now, and as long as you've accepted the license agreement and "license_accepted" is available, the installer will continue the installation process.
8. After the loading of images starts, wait for the installation to finish. You'll see the message: tml-installer | INFO: Finished Workspace clean-up.
9. Open a browser and start a Jenkins console to build your own images:

```
http://<localhost/IP>:8080
```

10. In the Jenkins page, start the build_docker job to start building your own images.

What to do next

After building your Docker images, follow the steps for [Deploying the Cluster](#).

Setting up HTTP Proxy Authentication in Installer Jenkins

Before you begin

The Installer is started

Procedure

1. Select the **Manage Jenkins** option.
2. Select the **Plugin Manager** option.
3. On the **Advanced** tab, select **HTTP Proxy Configuration**.

The screenshot shows the Jenkins 'HTTP Proxy Configuration' page with the 'Advanced' tab selected. The configuration fields are as follows:

Field	Value
Server	10.250.1.40
Port	3128
User name	squiduser
Password	Concealed
No Proxy Host	127.0.0.1, localhost

Buttons visible: 'Submit', 'Advanced...', 'Change Password'.

Add the following details:

User name, Password, and No Proxy Host

Note: The following error occurs during start up of the installer:

```
tml-installer | java.io.IOException: Unable to tunnel through proxy. Proxy returns
"HTTP/1.1 407 Proxy Authentication Required"
```

After you add HTTP proxy authentication information, the error is not displayed.

Managing the Installer

Starting the Installer

Run the following command to start Installer:

```
./start-installer.sh DOCKER_HOST_IP
```

**Note:**

When starting the installer, use the HOST IP of the Docker host. "Host" here means the host machine which is hosting your Docker container.

For example, if running directly from MacOS, use the IP of the Mac. If running a Docker machine, use the IP of the docker-machine (for example, docker-machine ip <machine-name>).

Note that "127.0.0.1" is a loopback address, which **cannot** be used.

Accepting the License

After the above command is issued for the first time, the following license message is shown:

You must accept the License agreement before proceeding. Press ENTER key to read the License. Press q to finish reading.

After pressing **ENTER**, the license is shown.

After reading the license, press "q". The following question is prompted:

Do you accept the license? (y/n)

Upon receiving answer "n/N", installer quits.

Upon receiving answer "y/Y", installer records the acceptance of license in file license_accepted.txt, you will not be required to read and accept license again.

Example contents of file license_accepted.txt:

```
License accepted on Sat Aug 24 14:10:31 PDT 2019
```

When Local Edition Installer is successfully started, it prints the following message:
tml-installer | tml-installer is started.

Stopping the Installer

Run the following command to stop the Installer:

```
./stop-installer.sh
```

Accessing the Installer

In a browser, type the following URL:

```
http://DOCKER_HOST_IP:8080
```

The Installer uses the Jenkins web console.

Securing the Installer

To secure the Installer, in Jenkins console, navigate to:

Jenkins > Manage Jenkins > Configure Global Security

The following is an example security configuration.

Configure Global Security

☒ Enable security ?

☐ Disable remember me

Access Control

Security Realm

☐ Delegate to servlet container ?

☒ Jenkins' own user database ?

☐ Allow users to sign up ?

☐ LDAP ?

☐ Unix user/group database ?

Authorization

☐ Anyone can do anything ?

☐ Legacy mode ?

☐ Logged-in users can do anything ?

☐ Matrix-based security ?

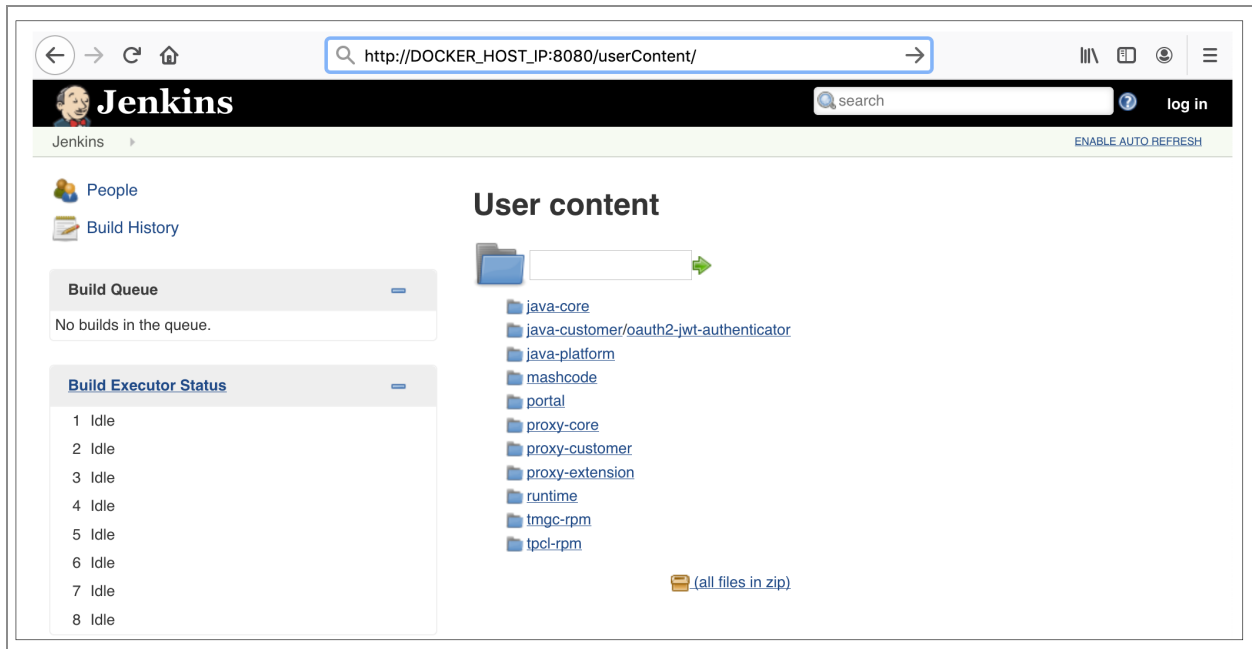
☒ Project-based Matrix Authorization Strategy ?

User/group	Overall	Credentials	Agent	Job	Run	View	SCM	Lockable Resources												
	Administer	Managed Domains	Build	Cancel	Configure	Create	Delete	Discover	Move	Read	Workspace	Replay	Update	Configure	Create	Delete	Read	Tag	Reserve	Unlock
Anonymous Users	☐	☒	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐
Authenticated Users	☒	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☒

Save **Apply**

The administrator can customize security configuration, but anonymous Overall Read access is needed for the Installer to build Docker images.

To ensure Overall Read access is granted to anonymous users, verify the following page is accessible without login:



Note: No confidential data is stored in the above location.

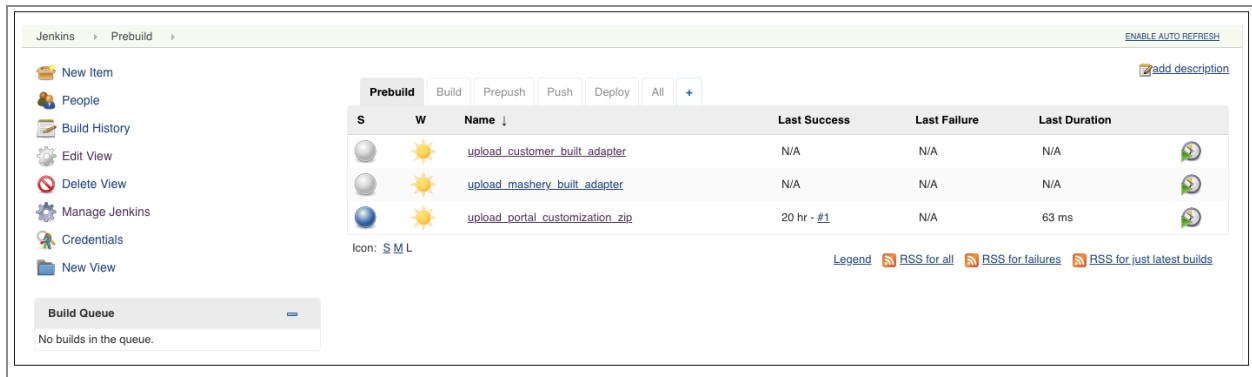
Note: After Jenkins is secured, Quick Start will stop working.

Prebuilding

If you have:

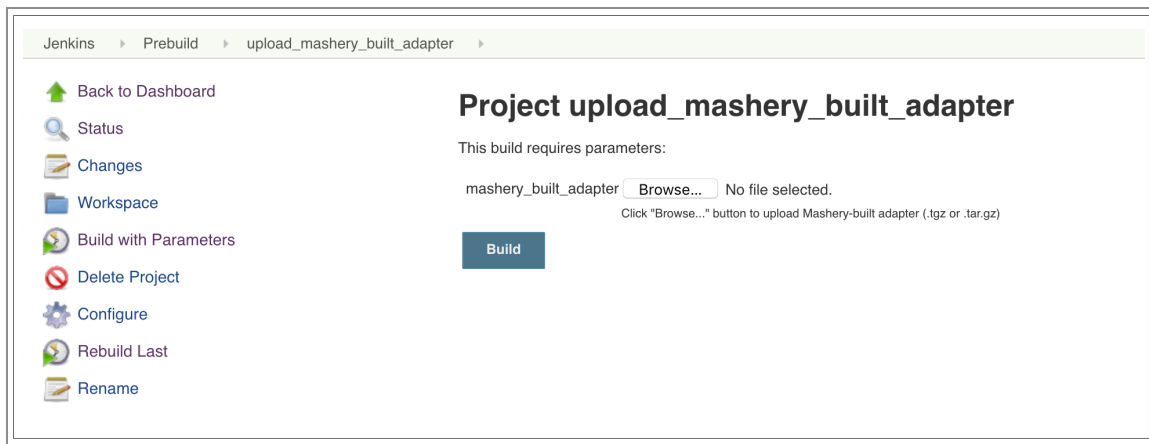
- [Boomi Cloud™ API Management-built connector or adapter](#)
- [Customer-built adapter from the SDK](#)
- [Portal customization package](#)

Then you can use one of the following Jenkins jobs to upload the contents before building the Docker images.



Uploading the Boomi Cloud™ API Management-built Connector or Adapter

As an Administrator user, the Jenkins job "upload_mashery_built_adapter" can be used to upload Boomi-built Connectors and adapters, and is accessed using the Jenkins web console.



Note that:

- A Boomi Cloud™ API Management-built connector or adapter is available to customers by using the Boomi Cloud™ API Management FTP service.
- A Boomi Cloud™ API Management-built connector or adapter is a tar.gz or tgz file.

Following is an example of the contents:

```
-rw-r--r-- artifacts.jar
drwxr-xr-x binary
-rw-r--r-- content.jar
-r-xr-x features
```

```
drwxr-xr-x plugins
```

The compressed contents of the Boomi Cloud™ API Management-built connector or adapter are extracted to `"/var/jenkins_home/userContent/proxy-customer"` in the tmgc-installer container. When the Docker image for Traffic Manager is built, this Boomi Cloud™ API Management-built connector or adapter is built into the Docker image.

For more information on how to get Boomi Cloud™ API Management built Connectors and Adapters tarball, refer to the [FAQs section](#).

Uploading the Customer-built Adapter

As an Administrator user, the Jenkins job "upload_customer_built_adapter" can be used to upload Customer-built adapters, and is accessed using the Jenkins web console.

The Jenkins job saves your adapter as:

```
/var/jenkins_home/userContent/proxy-extension/customer_extension.zip
```

To upload a Customer-built adapter:

- Click **Browse...** to select the connector or adapter file.
- Click **Build** to upload the connector or adapter to the Jenkins server.

Following is an example build output:

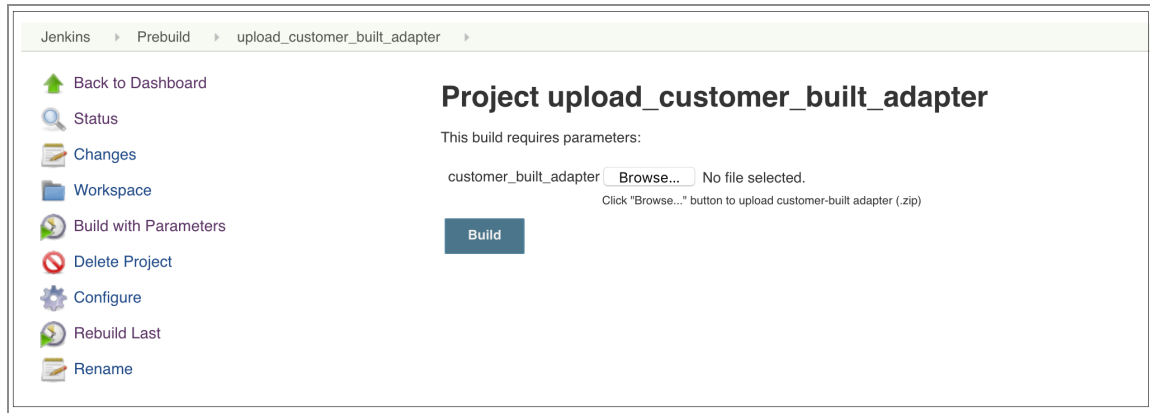
```
15:50:11 Updating /opt/javaproxy/proxy...
15:50:12 INFO Profile Local
15:50:12 INFO Timestamp Sat, 17 Nov 2018 23:50:11 GMT
15:50:12 INFO Found com.mashery.apiserver.oauth.feature.group 1.16.0.v20181116-0033
Mashery API Server
15:50:12 INFO Found com.mashery.proxy.local.customer.extensions.feature.group 1.0.0
Customer Extensions
15:50:12 INFO Found com.mashery.proxy.local.customer.feature.group 1.0.0 Customer-specific
Bundles
15:50:12 INFO Found com.mashery.proxy.local.product 1.46.0.v20181116-2143 Mashery Proxy
15:50:13 INFO Installing com.mashery.proxy.local.customer.feature.group 1.59.0.v20180813-
0728
15:50:13 INFO Profile updated.
15:50:13 INFO Timestamp Sat, 17 Nov 2018 23:50:13 GMT
15:50:13 INFO Found com.mashery.apiserver.oauth.feature.group 1.16.0.v20181116-0033
Mashery API Server
15:50:13 INFO Found com.mashery.proxy.local.customer.extensions.feature.group 1.0.0
Customer Extensions
```

```

15:50:13 INFO Found com.mashery.proxy.local.customer.feature.group 1.59.0.v20180813-0728
Customer-specific Bundles
15:50:13 INFO Found com.mashery.proxy.local.product 1.46.0.v20181116-2143 Mashery Proxy
15:50:14 INFO Operation completed in 1996 ms.

```

After the Docker image is built, the Customer-built adapter is built into the tml-ml Docker image.



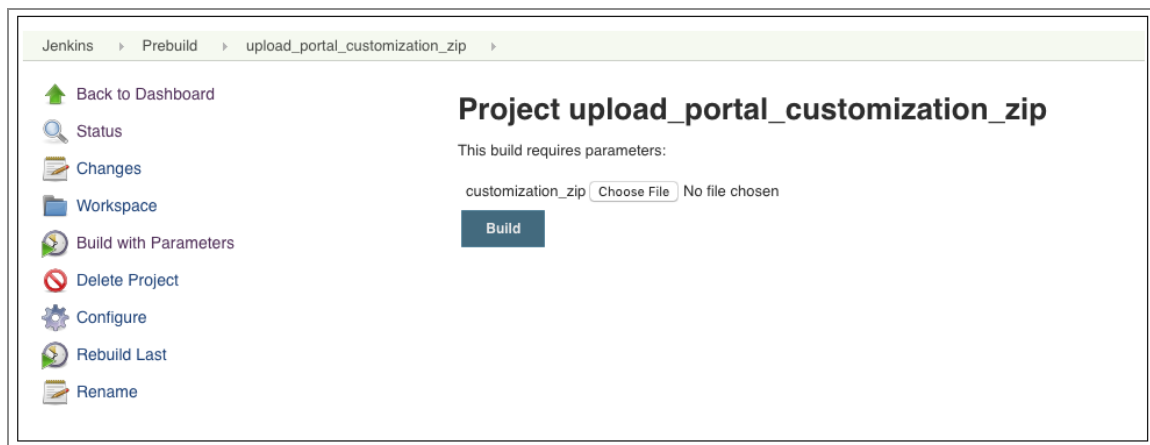
Uploading the Portal Customization Zip

As an Administrator user, upload the portal customization package in the zip.

The Jenkins job saves the customization package as:

```
/var/jenkins_home/userContent/portal/customization.zip
```

After the Docker image is built, the uploaded portal customization package is built into the tml-ml Docker image.



Building and Verifying the Docker Images

The Jenkins job `build_docker` is used to build the Docker images and to verify the images before proceeding to the next step.

**Note:**

- Third-party components required for image creation are packaged with the installer. Details on the packaged components are available in the License Agreement, enclosed with the installer bundle. Additional components, that are also required for the image creation process are documented here, must be downloaded during the image creation process from third-party sources.
- You can create and customize the docker image build process as needed, based on the required components outlined in the License Agreement and Third Party Components. As a convenience, for use or reference, Boomi has provided sample docker image build scripts to use out of the box. These scripts are available in `docker-build.tar.gz` in the installer bundle.

build_docker

The screenshot shows the Jenkins web interface for the 'build_docker_v3' project. The left sidebar contains navigation links: Back to Dashboard, Status, Changes, Workspace, Build with Parameters, Delete Project, Configure, Rebuild Last, and Rename. Below these is a 'Build History' section with a search bar and links to Atom feeds. The main configuration area is titled 'Project build_docker_v3' and includes the following fields and options:

- RELEASE_VERSION**: Input field with value '5.4.0'.
- RELEASE_SUFFIX**: Input field with value 'GA'.
- BUILD_DOCKER_NOSQL**: Checked checkbox. Build Cassandra docker image.
- BUILD_DOCKER_SQL**: Checked checkbox. Build MySQL docker image. (defaults to MySQL CE. For MySQL SE, please refer to documentation).
- BUILD_DOCKER_CM**: Checked checkbox. Build Cluster Manager docker image.
- BUILD_DOCKER_LOG**: Checked checkbox. Build log service docker image.
- BUILD_DOCKER_REPORTING**: Unchecked checkbox. Build Reporting docker image.
- BUILD_DOCKER_CACHE**: Checked checkbox. Build cache docker image.
- BUILD_DOCKER_TM**: Checked checkbox. Build Traffic Manager docker image.
- TM_CONNECTORS**: Radio button selected for 'Connectors available from the installer.' Other options are 'Connectors uploaded from "upload_mashery_built_adapters"' and 'None'.
- UPDATE_PACKAGES**: Dropdown menu set to 'Update_Specified_Packages'. Below it, text says 'Update all packages or specified packages when docker images are built.'
- update_package_list**: 'Browse...' button. Text says 'No file selected.' and '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**: Unchecked checkbox.

A blue 'Build' button is located at the bottom of the configuration area. The footer of the page indicates 'Page generated: Dec 16, 2020 11:01:28 PM GMT' and 'Jenkins 2.235'.

RELEASE_VERSION

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

RELEASE_SUFFIX

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

TM_CONNECTORS

You can choose from the following options for building a Local Edition Connector for the Traffic Manager container (tml-tm):

- **Connectors available from the installer** - loads the connectors that are readily available from the installer.
- **Connectors uploaded from "upload_mashery_built_adapters"** - only loads the connectors uploaded from the pre-build step.
- **None**



Note:

If you build with uploaded and then later build again with bundled, the later images will have connectors from the installer.

If you build with uploaded and then upload a new tar.gz in prebuild, then rebuild again with uploaded, the latest uploaded connectors are included.

If you select none, a 1.0 version of connector is installed. None is actually does not mean no connector is installed.

For more information on available options for building connectors, refer to [Building and Installing Connectors](#).

Choosing the Docker Images to Build

An Administrator builds the Local Edition Docker images using this Jenkins job:

Jenkins Job Name	Mashery Local Docker image
BUILD_DOCKER_NOSQL	tml-nosql: NoSQL container
BUILD_DOCKER_LOG	tml-log: Log container
BUILD_DOCKER_SQL	tml-sql: SQL container
BUILD_DOCKER_CACHE	tml-cache: Cache container
BUILD_DOCKER_TM	tml-tm: Traffic Manager container

Jenkins Job Name	Mashery Local Docker image
BUILD_DOCKER_CM	tml-cm: Cluster Manager container



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-lib
device-mapper.x86_64
device-mapper-lib.x86_64
```


VERIFY_DOCKER_IMAGE

Select the VERIFY_DOCKER_IMAGE checkbox to validate the images before carrying out the setup.

The VERIFY_DOCKER_IMAGE option checks the following:

- Installed packages,
- Contents of the Local Edition directories and files,
- Files required by Local Edition scripts and services.

Building and Installing Local Edition Connectors

The following options are available for building and installing Local Edition Connectors:

- Build the connectors to be bundled in the Local Edition 5.4.0 Installer.
- Build the connectors to be uploaded from Installer Jenkins pre-build step.
- Build the Installer with the bundle connectors.

Building the Bundled Connectors

Scenario 1: Customer requests all Boomi Cloud™ API Management built Adapters Only

- Boomi will provide Adapters certified for Local Edition.
- Boomi will certify Adapters for Docker with Kubernetes.
- Boomi will not provide further certification required for a Boomi Cloud™ API Management built adapter for later versions of Local Edition 5.X.X as long as Boomi Cloud™ API Management maintains compatibility.



Note: Exception case:

- There is an Adapter update which was requested either by a customer or due to a defect found in Adapter.

Scenario 2: Customer requests all Boomi Cloud™ API Management built Adapters along with all Boomi Cloud™ API Management Ported Connectors.

- Boomi will provide Adapters and Boomi Cloud™ API Management ported

Connectors certified for Local Edition.

- Boomi will certify Adapters and Boomi Cloud™ API Management ported Connectors for Docker with Kubernetes.
- Boomi will not provide further certification for Boomi Cloud™ API Management built adapters or ported Connectors for later versions of Local Edition 5.X.X as long as Boomi Cloud™ API Management maintains compatibility.



Note: Exception case:

- There is an Adapter update which was requested either by customer or defect found in Adapter.
- There is update in Connector or new Adapter is production released and requested by Customer.

Building the Connectors to Upload

Scenario 1: Customer requests all Boomi Cloud™ API Management Ported Connectors:

- Boomi will provide Boomi Cloud™ API Management ported Connectors certified for Local Edition.
- Boomi will certify Boomi Cloud™ API Management ported Connectors for Docker with Kubernetes.
- Boomi will not provide further certification for Boomi Cloud™ API Management ported Connectors for later version of Local Edition 5.X.X as long as Traffic Manager product maintains compatibility.



Note: Exception case:

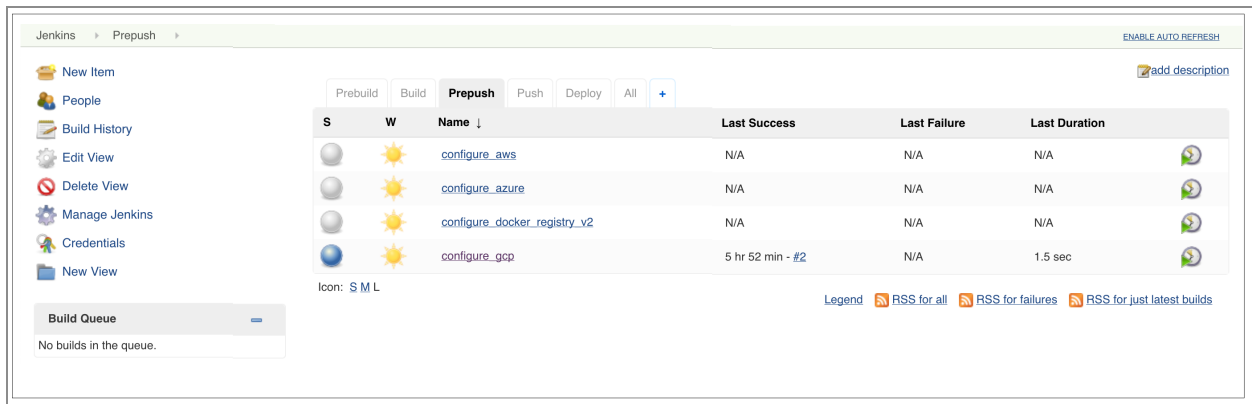
- There is update in a Connector or a new Connector is production released and requested by Customer.

Building the Installer with the Bundled Connectors

Build the installer as needed with the Local Edition builder's "tml_build_installer".

Prepushing

You can configure the access to supported platforms before pushing the Local Edition Docker images and starting Local Edition cluster deployment.



The access to the following platforms and container registries can be configured:

Jenkins Job Name	Container Registry	Platform
configure_aws	AWS ECR (AWS Elastic Container Registry)	AWS
configure_azure	Azure Container Registry	Azure
configure_docker_registry_v2	Docker Registry V2	Docker Swarm, etc.
configure_gcp	GCP Container Registry	GCP (Google Cloud Platform)

Configuring AWS

For deployment of Local Edition cluster in Kubernetes at AWS, it is recommended to use Access Key / Secret Access Key from service account. AWS IAM role is also supported.

Access Key / Secret Access Key / AWS IAM role are used to:

- create AWS Container Registry;
- push Local Edition Docker images to AWS Container Registry;

- create Kubernetes cluster in AWS;
- create Local Edition cluster in Kubernetes at AWS.

For deployment details, refer to [AWS using Kubernetes](#).

Name	Definition
aws_access_key_id	AWS Access Key ID
aws_secret_access_key	AWS Secret Access Key
aws_default_region	AWS default region
aws_role_arn	AWS role arn

Configuring Azure

For deployment of Local Edition cluster in Kubernetes in Azure, it is recommended to use Service Principal. Service Principal is used to:

- Create Azure Container Registry;
- Push Local Edition Docker images to Azure Container Registry;
- Create Kubernetes cluster using Azure Kubernetes Service;
- Create Local Edition cluster in Kubernetes at Azure.

For deployment details, refer to [Azure using Kubernetes](#).

Jenkins > Prepush > configure_azure >

[Back to Dashboard](#)

[Status](#)

[Changes](#)

[Workspace](#)

[Build with Parameters](#)

[Delete Project](#)

[Configure](#)

[Rebuild Last](#)

[Rename](#)

Project configure_azure

This build requires parameters:

DIRECTORY_ID

Directory (tenant) ID

APPLICATION_ID

Application (client) ID

CLIENT_SECRET

Client Secret

RESOURCE_GROUP

RESOURCE_GROUP_REGION

CONTAINER_REGISTRY

[Build](#)

Build History [trend](#)

[RSS for all](#) [RSS for failures](#)

Name	Definition	Notes
DIRECTORY_ID	Directory (tenant) ID	
APPLICATION_ID	Application (client) ID	
CLIENT_SECRET	Client Secret	
RESOURCE_GROUP	Resource Group	The resource group will be created if it does not exist.
RESOURCE_GROUP_REGION	Resource Group Region	
CONTAINER_REGISTRY	Container Registry	The container registry will be created if it does not exist.

Configuring CGP

For deployment of Local Edition cluster in Kubernetes at GCP, it is recommended to use Service Account. Service Account is used to:

- Create GCP Container Registry;
- Push Local Edition Docker images to GCP Container Registry;
- Create Kubernetes cluster using GCP Kubernetes Service;
- Create Local Edition cluster in Kubernetes at GCP.

For deployment details, refer to [Deployment on Google Cloud Platform using Kubernetes](#).

Name	Definition	Notes
GCP_REGISTRY_HOST	GCP Registry Host	These are currently available for different regions (https://cloud.google.com/container-registry/docs/pushing-and-pulling): • gcr.io • us.gcr.io • eu.gcr.io • asia.gcr.io
GCP_REGISTRY_HOST	GCP Registry Host	
service_account_json_key_file	The JSON file contains keys for the service account	

Configure Docker Registry v2

For deployment on Docker Swarm, Local Edition Docker images can be stored in Docker Registry v2.

Jenkins > Prepush > configure_docker_registry_v2

Project configure_docker_registry_v2

This build requires parameters:

DOCKER_REGISTRY_HOST

IP or hostname of docker registry host

DOCKER_REGISTRY_PORT

DOCKER_REGISTRY_USERNAME

if username / password is required for authentication to docker registry.

DOCKER_REGISTRY_PASSWORD

DOCKER_REGISTRY_REPO

Build

Build History [trend](#)

find x

Name	Definition
DOCKER_REGISTRY_HOST	Docker Registry Host
DOCKER_REGISTRY_PORT	Docker Registry Port
DOCKER_REGISTRY_USERNAME	Docker Registry Username
DOCKER_REGISTRY_PASSWORD	Docker Registry Password
DOCKER_REGISTRY_REPO	Docker Registry Repo

Pushing the Docker Images

After the access to the supported platform is configured, the administrator can push Local Edition Docker images to the supported Docker registry.

The screenshot shows the Jenkins 'Push' view. On the left, there's a sidebar with navigation links: New Item, People, Build History, Edit View, Delete View, Manage Jenkins, Credentials, and New View. Below these are sections for 'Build Queue' (No builds in the queue) and 'Build Executor Status' (4 idle executors). The main area displays a table of jobs under the 'Push' tab. The table has columns for status (S), icon (W), name, last success, last failure, and last duration. Jobs include 'push_docker_to_aws', 'push_docker_to_aws_helper', 'push_docker_to_azure', 'push_docker_to_azure_helper', 'push_docker_to_docker_registry_v2', 'push_docker_to_docker_registry_v2_helper', 'push_docker_to_gcp', 'push_docker_to_gcp_helper', and 'save_docker_to_archive'. The 'push_docker_to_gcp' job shows a successful build 12 hours ago. At the bottom, there's a legend for RSS feeds: RSS for all, RSS for failures, and RSS for just latest builds.

Supported Platform / Docker Registry	Main Jenkins Job	Helper Jenkins Job
AWS	push_docker_to_aws	push_docker_to_aws_helper
Azure	push_docker_to_azure	push_docker_to_azure_helper
GCP	push_docker_to_gcp	push_docker_to_gcp_helper
Docker Registry V2	push_docker_to_docker_registry_v2	push_docker_to_docker_registry_v2_helper

**Note:**

- Use main Jenkins job, do not use helper Jenkins job directly.
- Only the current successful build of Local Edition Docker images are available to push; each new build of Local Edition Docker images will first erase Docker images from previous builds.

The parameters required to push Local Edition Docker images to supported platforms are the same:

Name	Definition	Notes

DOCKER_TAG	docker image tag	Docker image tag starts with 'v', for example, "v5.2.0.1"
PUSH_DOCKER_NOSQL	tml-nosql	
PUSH_DOCKER_LOG	tml-log	
PUSH_DOCKER_SQL	tml-sql	
PUSH_DOCKER_CACHE	tml-cache	
PUSH_DOCKER_TM	tml-tm	
PUSH_DOCKER_CM	tml-cm	

Pushing Docker to AWS

Jenkins
Push
push_docker_to_aws

Back to Dashboard
 Status
 Changes
 Build with Parameters
 Delete Pipeline
 Configure
 Full Stage View
 Rename
 Pipeline Syntax

Build History trend

Pipeline push_docker_to_aws

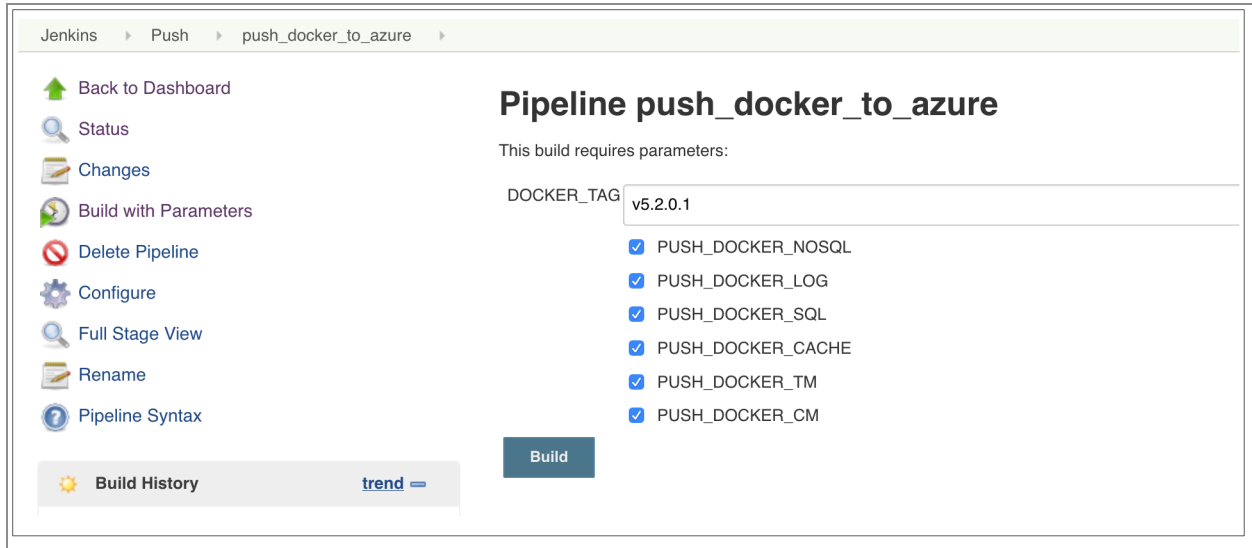
This build requires parameters:

DOCKER_TAG

- ☒ PUSH_DOCKER_NOSQL
- ☒ PUSH_DOCKER_LOG
- ☒ PUSH_DOCKER_SQL
- ☒ PUSH_DOCKER_CACHE
- ☒ PUSH_DOCKER_TM
- ☒ PUSH_DOCKER_CM

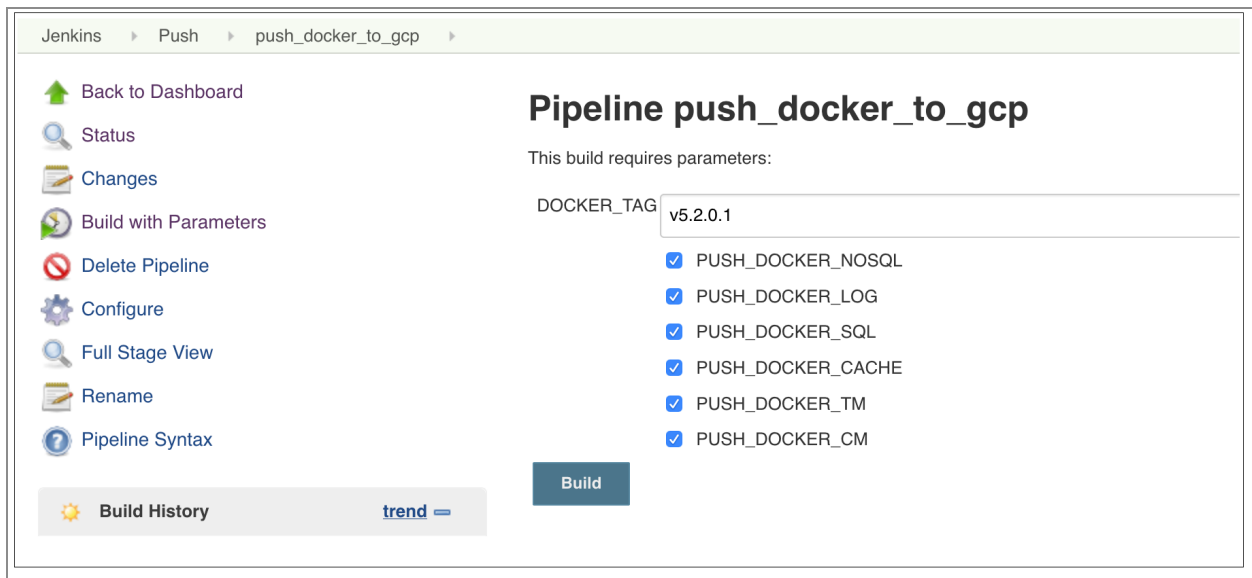
Build

Pushing Docker to Azure



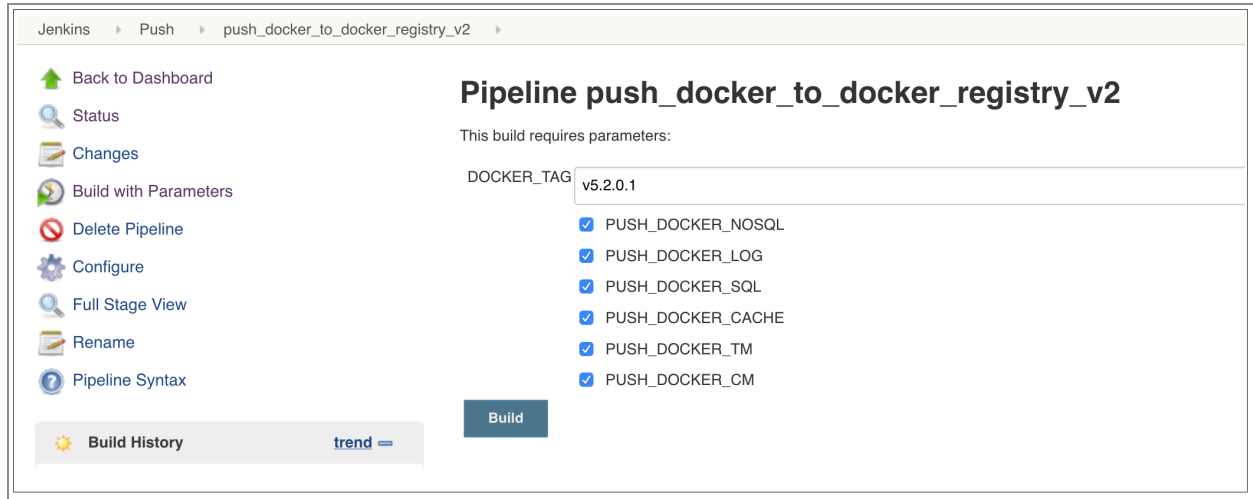
The screenshot shows the Jenkins web interface for configuring a pipeline named 'push_docker_to_azure'. The breadcrumb trail at the top reads 'Jenkins > Push > push_docker_to_azure'. On the left sidebar, there are links for 'Back to Dashboard', 'Status', 'Changes', 'Build with Parameters', 'Delete Pipeline', 'Configure', 'Full Stage View', 'Rename', and 'Pipeline Syntax'. At the bottom of the sidebar is a 'Build History' section with a 'trend' link. The main content area is titled 'Pipeline push_docker_to_azure' and includes the text 'This build requires parameters:'. Below this, there is a 'DOCKER_TAG' parameter set to 'v5.2.0.1'. A list of six checkboxes is shown, all of which are checked: 'PUSH_DOCKER_NOSQL', 'PUSH_DOCKER_LOG', 'PUSH_DOCKER_SQL', 'PUSH_DOCKER_CACHE', 'PUSH_DOCKER_TM', and 'PUSH_DOCKER_CM'. A blue 'Build' button is located at the bottom right of the configuration area.

Pushing Docker to GCP



The screenshot shows the Jenkins web interface for configuring a pipeline named 'push_docker_to_gcp'. The breadcrumb trail at the top reads 'Jenkins > Push > push_docker_to_gcp'. The left sidebar and 'Build History' section are identical to the previous screenshot. The main content area is titled 'Pipeline push_docker_to_gcp' and includes the text 'This build requires parameters:'. Below this, there is a 'DOCKER_TAG' parameter set to 'v5.2.0.1'. A list of six checkboxes is shown, all of which are checked: 'PUSH_DOCKER_NOSQL', 'PUSH_DOCKER_LOG', 'PUSH_DOCKER_SQL', 'PUSH_DOCKER_CACHE', 'PUSH_DOCKER_TM', and 'PUSH_DOCKER_CM'. A blue 'Build' button is located at the bottom right of the configuration area.

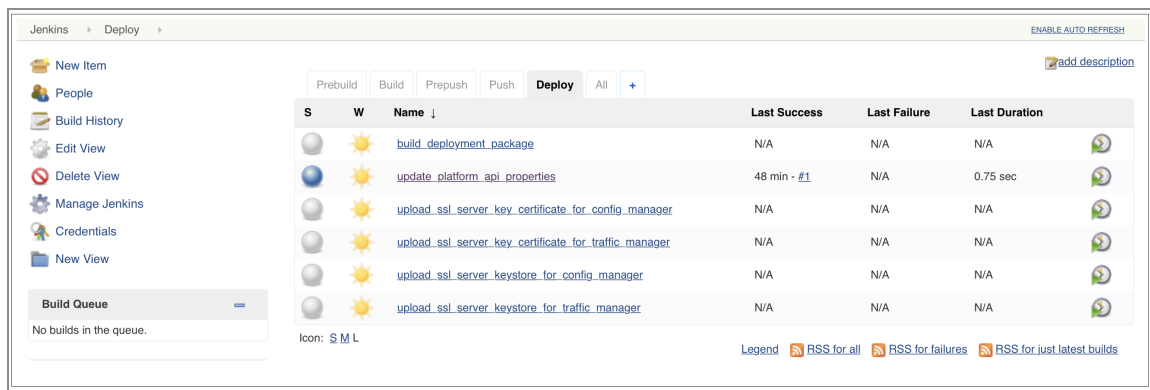
Pushing Docker to Docker Registry v2



Deploying the Local Edition Cluster

To deploy the Local Edition cluster, the administrator should:

- Update platform API (V3) properties (Jenkins job `update_platform_api_properties`),
- Customize key stores for Traffic Manager and Configuration Manager (Jenkins jobs whose names starting with `upload_ssl_*`).



Updating Platform API Properties

For security, the administrator should update UUIDs for areas, update package keys and secrets.

Jenkins > Deploy > update_platform_api_properties >

Project update_platform_api_properties

This build requires parameters:

DOMAIN_SUFFIX
Your domain suffix.

AREA_NAME
Your area name.

☒ update_uuids_for_areas
Generate new UUIDs for areas.

☒ update_package_keys
Generate new keys for packages.

☒ update_package_secrets
Generate new secrets for packages.

Build History [trend](#)

Build

Using this Jenkins job, the administrator can:

- Customize domain suffix
- Customize area name
- Generate new UUIDs for areas
- Generate new keys for packages
- Generate new secrets for packages

The file `/var/jenkins_home/docker-deploy/properties/tml_papi_properties.json` is updated.

Prepare Keystore for Traffic Manager

A keystore for Traffic Manager can be prepared using either of the following Jenkins jobs:

- **Uploading SSL Server Keystore for Traffic Manager**

Jenkins > Deploy > upload_ssl_server_keystore_for_traffic_manager >

Project upload_ssl_server_keystore_for_traffic_manager

This build requires parameters:

keystore_file No file chosen
SSL keystore file in PKCS#12 format.

keystore_password
SSL keystore password

Build

To upload an SSL server keystore for Traffic Manager:

1. Upload the keystore in PKCS#12 format.
2. Input the password protecting the keystore.

The Jenkins job delivers the following output:

- Converts the keystore to JKS format from PKCS#12 format and saves it as `/var/jenkins_home/docker-deploy/properties/tml-tm.jks`.
 - Records the keystore password in `tm_keystore_password` in `/var/jenkins_home/docker-deploy/properties/tml_tm_properties.json`.
- **Uploading SSL Server Key Certificate Traffic Manager**

To upload an SSL server key certificate for Traffic Manager:

1. Upload the certificate in PEM format.
2. Upload the private key in PKCS#8 format and input the password protecting the private key.

The Jenkins job delivers the following output:

- Puts the certificate and private key to keystore in JKS format and saves it as `/var/jenkins_home/docker-deploy/properties/tml-cm.jks`.
- Records the keystore password in `cm_keystore_password` in `/var/jenkins_home/docker-deploy/properties/tml_cm_properties.json`.

Prepare Keystore for Configuration Manager

A keystore for Traffic Manager can be prepared using either of the following Jenkins jobs:

- **Uploading SSL server keystore for Configuration Manager**

The screenshot shows the Jenkins job configuration page for the project **upload_ssl_server_keystore_for_config_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 displays the job title and a message: "This build requires parameters:". Below this, there are two parameters: **keystore_file** with a "Choose File" button and the text "No file chosen" and "SSL keystore file in PKCS#12 format.", and **keystore_password** with a password input field and the text "SSL keystore password". A blue "Build" button is located below the parameters.

To upload an SSL server keystore for Configuration Manager:

1. Upload the keystore in PKCS#12 format.
2. Input the password protecting the keystore.

The Jenkins job delivers the following output:

- Converts the keystore to JKS format from PKCS#12 format and saves it as `/var/jenkins_home/docker-deploy/properties/tml-tm.jks`.
- Exports certificate as `/var/jenkins_home/docker-deploy/properties/tml-cm-crt.pem`.
- Exports private key as `/var/jenkins_home/docker-deploy/properties/tml-cm-key.pk8`.
- Records the keystore password in `cm_keystore_password` in `/var/jenkins_home/docker-deploy/properties/tml_tm_properties.json`.

• Uploading SSL Server Key Certificate for Configuration Manager

The screenshot shows the Jenkins job configuration page for the project **upload_ssl_server_key_certificate_for_config_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 displays the job title and a message: "This build requires parameters:". Below this, there are three parameters: **certificate_file** with a "Choose File" button and the text "No file chosen" and "SSL certificate file.", **key_file** with a "Choose File" button and the text "No file chosen" and "Private key file in PKCS#8 format.", and **key_password** with a password input field and the text "Password protecting key file.". A blue "Build" button is located below the parameters.

To upload an SSL server key certificate for Configuration Manager:

1. Upload the certificate in PEM format.

2. Upload the private key in PKCS#8 format and input the password protecting the private key.

The Jenkins job delivers the following output:

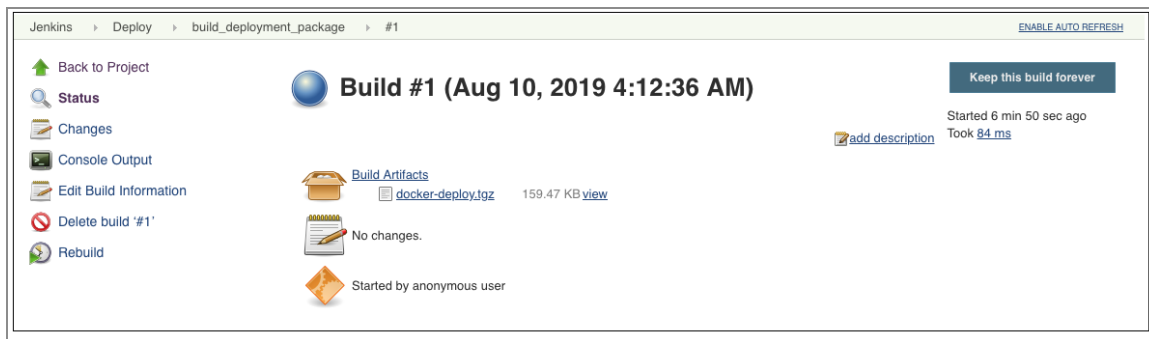
- Saves the certificate as `/var/jenkins_home/docker-deploy/properties/tml-cm-crt.pem`.
- Saves the private key as `/var/jenkins_home/docker-deploy/properties/tml-cm-key.pk8`.
- Puts the certificate and private key to keystore in JKS format and saves it as `/var/jenkins_home/docker-deploy/properties/tml-cm.jks`.
- Records the keystore password in `cm_keystore_password` in `/var/jenkins_home/docker-deploy/properties/tml_cm_properties.json`.

Building the Deployment Package

The administrator can:

- Deploy the Local Edition cluster inside the Installer,
- or build the deployment package, download the deployment package, and deploy the Local Edition cluster from another host.

Click **Build Now** in Jenkins job `build_deployment_package` to build the deployment package. The deployment package **docker-deploy.tgz** is available for download from the build archive:



Note: The keystores, keys and certificates are included the deployment package.

To start deployment, refer to the deployment pages for the supported platform.

Deploying a Local Edition Cluster

Local Edition leverages Kubernetes, Docker Swarm, or OpenShift for container orchestration and has been tested on AWS, Azure, and GCP cloud platforms, as well as on-premises via bare metal RHEL, CentOS, and VMWare virtualized environments.

In the tethered mode MySQL container and Log service container can communicate with MOM server.

- MySQL container runs the On-prem loader (OPL) and it is the OPL that synchronizes API configuration data between cloud control center and Local Edition on-prem. MySQL thereby communicates with the MOM server to down-sync the configuration data. It also synchronizes any data pertaining to the identity and certificates and https client profile with up and down synchronization.
- Log service container that runs the Access-log consumer (ALC) and ALC synchronizes the access log data between the MOM and Local Edition on-prem. Log service thereby communicates with the MOM server to up-sync the access log data.

See [Platform and Deployment Options](#) for more information.

For deployment sequence using custom scripts or any combined practices of Continuous Integration (CI) and Continuous Delivery (CD) refer [Deployment Sequence](#) topic.

Deployment Topologies

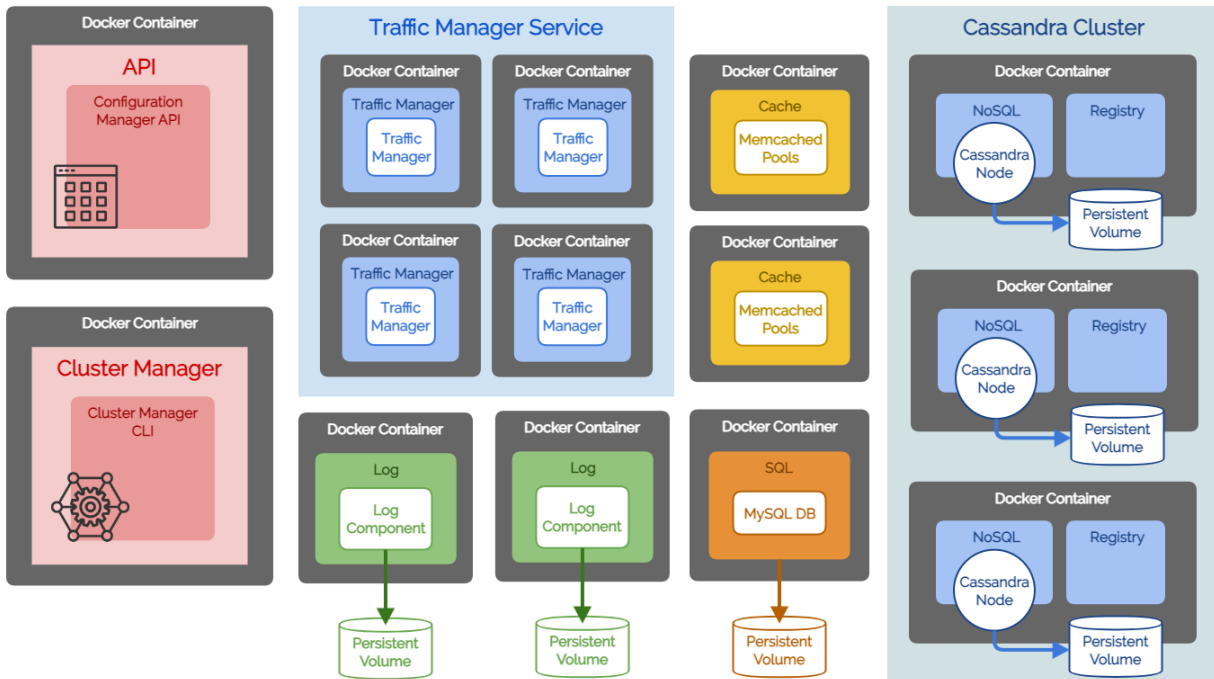
- [Docker Container View of a Single Zone](#)
- [Kubernetes View of a Single Zone Cluster](#)
- [Kubernetes View of a Multizone Cluster](#)

i Note: The StatefulSet behavior in Kubernetes provides expected resiliency. However, Docker swarm implementation requires manual remediation and resiliency is NOT automatic with Local Edition's Swarm deployment.

Docker Container View of a Single Zone

Docker Container View of a Single Zone

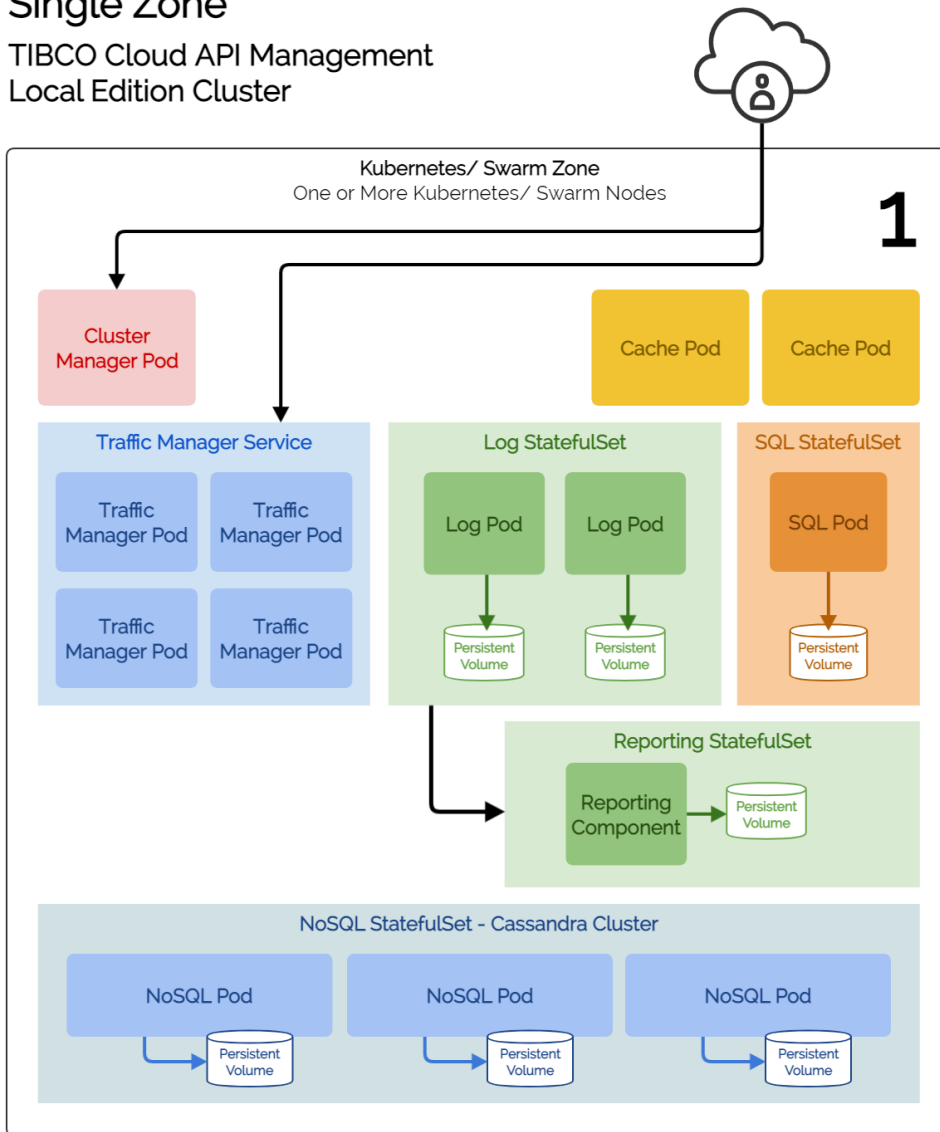
TIBCO Cloud API Management Local Edition Cluster
(One or more machine instances)



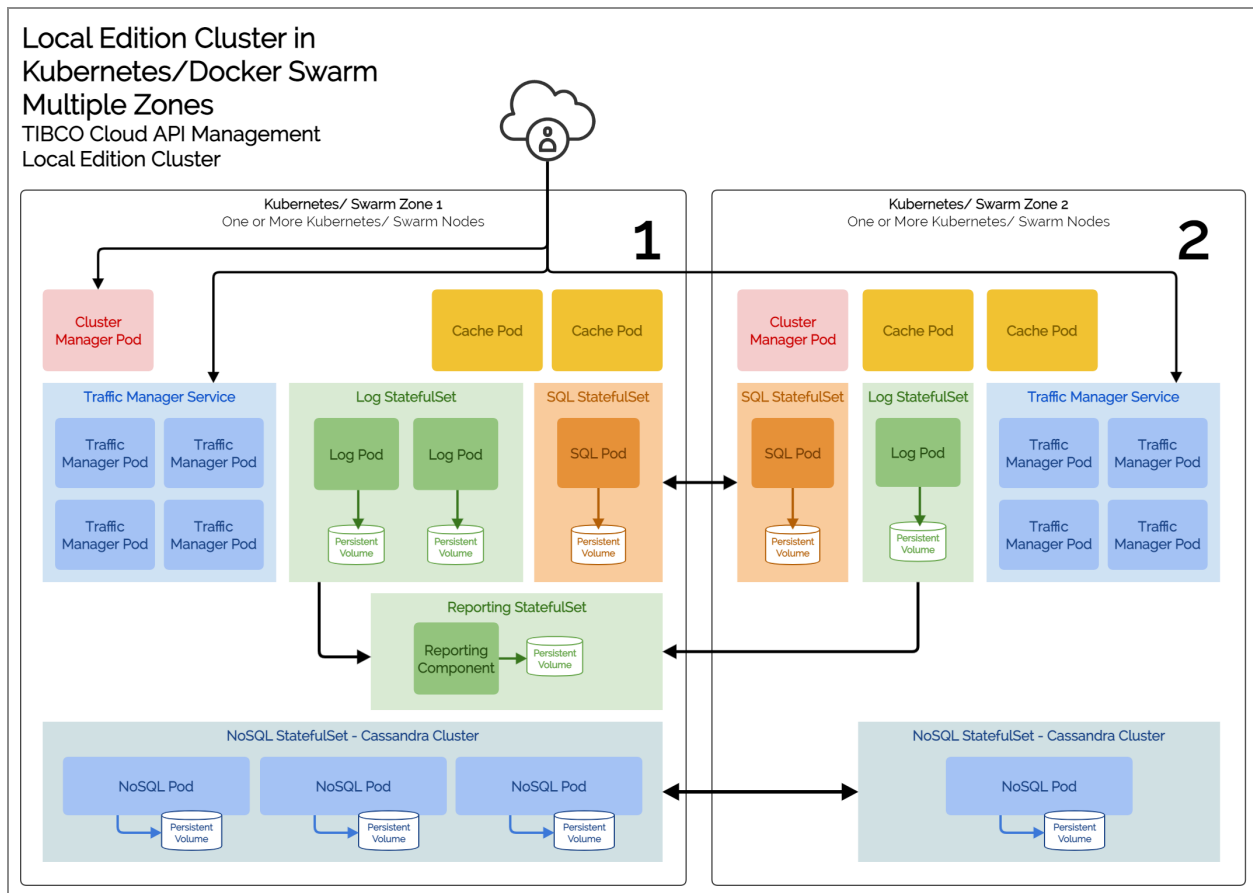
Local Edition Cluster in Kubernetes / Docker Swarm - Single Zone

Local Edition Cluster in Kubernetes/Docker Swarm Single Zone

TIBCO Cloud API Management
Local Edition Cluster



Local Edition Cluster in Kubernetes Docker Swarm - Multizone



i Note: In a multi-zone deployment, the quota limit and/or throttle limit should be divided among zones. For example, in a three-zone Local Edition deployment, with a quota limit of 90 in total, 30 should be defined in each zone.

User Content, Templates and Resources

You can provision custom resources and user content within each specific pod.

To upload custom resources copy the required files under `docker-deploy/resources/tml-<Type>`. These files will be available inside the pod, post deployment at `/opt/mashery/resources`.

While deploying a Local Edition cluster, the resources inside each tml-type folder is uploaded as a secret inside the pod.

Map the tml-type as follows:

TML-Type	Secret Name	Description
tml-cm	cm-resource	All configuration manager pods
tml-log	log-resource	All log service pods
tml-tm	tm-resource	All traffic managers pods
tml-cache	cache-resource	All cache pods
tml-sql	sql-resource	All SQL pods
tml-nosql	nosql-resource	All NoSQL pods
tml-reporting	reporting-resource	Reporting pod

For kubernetes platforms, the resources uploaded cannot be modified inside the pod. To modify resources post deployment, recreate and apply the secret. For more information, see [kubernetes secrets](#).

For Docker swarm, you can modify the resources inside the container post deployment.

- [User Template](#)
- [Example](#)
- [FAQ](#)

User Templates

The custom adapter must contact an external service. For example, LDAP server and JMS services.

- The external resource configuration is common to all end points and pre and post inputs.

- If the custom configuration file is not available, you must replicate the configuration in all endpoint as pre and post inputs.
- Container/pod specific resources must be placed in specific directories.
- If a file or resource is required in more than one container, a copy must exist in directories meant for every container.
- Input the value for place holders in the templates of the respective container type properties.
- Before bringing up the cluster, copy the custom resources or configuration templates to the specific directories at <extract directory>/docker-deploy/resources/<tml-type>.
- The resources for each of the containers must be accompanied by a specification file in json format. The json file has to be named usertemplate_config.json. If user has content for more than one container type, each container type should have the spec file.

The structure of the usertemplate_config.json must be as follows.

```
{
  "userContent": [
    {
      "file": "a.json.template",
      "destination": "/etc/"
    },
    {
      "file": "b.json.template",
      "destination": "/opt/mydata/"
    },
    {
      "file": "c.properties.template",
      "destination": "/opt/mashery/resources/"
    },
    {
      "file": "d.xml.template",
      "destination": "/var/data/"
    }
  ]
}
```

Note: For td-agent templates in log container, specification file is not required. The template is used as configuration file.

Example

Following is a sample custom adapter configuration for tml-tm pod and authenticates requests against LDAP.

Before you begin

A configuration of an external LDAP system must to be available.

Procedure

1. Name the configuration file as `ldap.properties.template`.

ldap.properties.template

```
ldap.provider=${ldap_provider_url}
ldap.default.org=${ldap_default_org}
request.header.username.field=${request_field_username}
request.header.cred.field=${request_field_passwd}
```

2. Copy the template file at `<extract_location>/docker-deploy/resources/tml-tm` directory.
3. Edit `<extract_location>/docker-deploy/properties/tml_tm_properties.json` and input the value as follows.

tml_tm_properties.json new properties

```
"ldap_provider_url": "ldap://ldapservicename:389",
"ldap_default_org": "sampleorg",
"request_field_username": "username",
"request_field_passwd": "creds"
```

4. Define the user contents specification file `usertemplate_config.json`.

```
{
  "userContent": [
    {
      "file": "ldap.properties.template",
      "destination": "/etc/"
    }
  ]
}
```

5. Copy the `usertemplate_config.json` file to `<extract_location>/docker-deploy/resources/tml-tm/` and deploy the cluster.

FAQ

1. What is the syntax for custom configuration templates?

Template must be as per Apache Velocity 1.7 template rules and syntax. For more information, see <https://velocity.apache.org/engine/1.7/user-guide.html>.

2. Are there restrictions on destination directories?

Yes, certain directories are non-writable by some processes. Recommended locations are `/etc` or `/var` and their sub directories. Do not use `/opt/mashery` and sub directories as a destination since they are a read-only resource locations.

3. What happens if destination directory not exist?

The container agent will attempt to create the required directories, provided it has the required permissions. Check container agent logs for `IOExceptions`.

4. Can I provide any names for the templates?

Yes. the templates can have any name but the extension has to be `.template`. For example the template for `adapter.properties` will be named as `adapter.properties.template`.

5. Is the specification file mandatory?

Yes. If there is no specification file, the container agent will ignore the resources when merging configuration values and generating configurations.

6. Can I change custom configuration properties post cluster creation?

Yes. Using the `clustermanager cli clustermanager set the configuration` command. The container agent will regenerate configurations from the templates and new properties.

Configuring Properties Common to All Deployments

The administrator can configure properties common to all deployments.

 **Important:** These properties must be updated **before** running `compose.sh`.

In the Installer, property files are located at `/var/jenkins_home/docker-deploy/properties`.

For built and downloaded deployment packages, property files are at folder `docker-deploy/properties` after the deployment package is extracted.

Here is an example list of contents:

```
tml-cm-crt.pem
tml-cm-key.pk8
tml-cm.jks
tml-tm.jks
tml_cache_properties.json
tml_cluster_properties.json
```



```

tml_cm_properties.json
tml_log_properties.json
tml_nosql_properties.json
tml_papi_properties.json
tml_sql_properties.json
tml_tm_properties.json
tml_zones_properties.json

```

Keystores, Keys, and Certificates

File	Definition	Note
tml-cm-crt.pem	Certificate used in tml-cm	
tml-cm-key.pk8	Key used in tml-cm	
tml-cm.jks	Keystore used in tml-cm	tml-cm.jks has tml-cm-crt.pem and tml-cm-key.pk8 stored inside.

Customers should prepare their own key, certificate and keystore to be used in tml-cm.

The built-in key, certificate and keystore should not be used for production.

Refer to the *Preparing Keystore for Configuration Manager* section in [Deploying the Installer](#).

File	Definition
tml-tm.jks	Keystore used in tml-tm

Customers should prepare their own keystore to be used in tml-tm.

The built-in keystore should not be used for production.

Refer to the *Preparing Keystore for Traffic Manager* section in [Deploying the Installer](#).

Property Hierarchy

Cluster-Level Properties

File `tml_cluster_properties.json` specifies properties common to Local Edition cluster. Here are the default contents:

```
{
  "mysql_root_pwd": "changeme",
  "mysql_masheryonprem_pwd": "changeme",
  "mysql_mashonpremrepl_pwd": "changeme",
  "mysql_mashclient_pwd": "changeme",
  "mysql_masherybackup_pwd": "changeme",
  "mom_server": "https://api-mom.mashery.com",
  "mom_key": "",
  "mom_secret": ""
}
```

The administrator can customize all MySQL passwords:

```
"mysql_root_pwd": "changeme",
"mysql_masheryonprem_pwd": "changeme",
"mysql_mashonpremrepl_pwd": "changeme",
"mysql_mashclient_pwd": "changeme",
"mysql_masherybackup_pwd": "changeme",
```

The administrator can customize MOM access from Local Edition cluster in tethered mode:

```
"mom_server": "https://api-mom.mashery.com",
"mom_key": "",
"mom_secret": ""
```

Zone-Level Properties

File `tml_zones_properties.json` specifies properties per zone. Here are the default contents:

```
[
  {
    "zone_serial": "0",
    "mom_server_proxy_host": "",
    "mom_server_proxy_port": 3128,
    "mom_server_proxy_username": "",
```

```

    "mom_server_proxy_password": "",
    "target_server_proxy_host": "",
    "target_server_proxy_port": 3128,
    "no_proxy_pool": ""
  },
  {
    "zone_serial": "1",
    "mom_server_proxy_host": "",
    "mom_server_proxy_port": 3128,
    "mom_server_proxy_username": "",
    "mom_server_proxy_password": "",
    "target_server_proxy_host": "",
    "target_server_proxy_port": 3128,
    "no_proxy_pool": ""
  },
  {
    "zone_serial": "2",
    "mom_server_proxy_host": "",
    "mom_server_proxy_port": 3128,
    "mom_server_proxy_username": "",
    "mom_server_proxy_password": "",
    "target_server_proxy_host": "",
    "target_server_proxy_port": 3128,
    "no_proxy_pool": ""
  }
]

```

The administrator can customize HTTP Proxy for MOM access per zone:

```

"mom_server_proxy_host": "",
"mom_server_proxy_port": 3128,
"mom_server_proxy_username": "",
"mom_server_proxy_password": "",

```

The administrator can customize HTTP Proxy for accessing target servers per zone:

```

"target_server_proxy_host": "",
"target_server_proxy_port": 3128,
"no_proxy_pool": ""

```

**Note:**

Each zone is identified by `zone_serial`. The customer administrator can tailor the property JSON according the number of zones to which Local Edition cluster will be deployed.

Zone properties for zone whose `zone_serial` is above the actual number of deployed zones will not be used.

no_proxy_pool

For target servers covered in `no_proxy_pool`, Local Edition directly calls them without going through HTTP proxy.

REGEX (Regular Expression) can be use in the property value.

Example #1:



Note: Note the backslash for escaping double quote.

```
"no_proxy_pool": "\"target1.na.tibco.com\", \"target2.na.tibco.com\""
```

In this example, if the target server is `target1.na.tibco.com` or `target2.na.tibco.com`, HTTP proxy server will be bypassed.

Example #2:



Note: Note the backslash for escaping double quote.

```
"no_proxy_pool": "\"back.*.na.tibco.com\""
```

In this example, `back.*.na.tibco.com` is a regular expression, it matches `back1.na.tibco.com` and `back2.na.tibco.com`.

Platform API Properties

The `tml_papi_properties.json` file specifies properties for platform API V3.

Here are the default properties:

```
{
  "sp_config_environment_domain_suffix": "mashery.com",
  "areas_customer_area_name": "tml",
  "localdevadmin_handle": "admin",
  "localdevadmin_password": "Ap1Us3rPasswd",
  "masherytestautomation_handle": "mltestautomation",
  "area_uuids": [
    {
      "area_uuid_name": "areas_customer_area_uuid",
      "area_uuid_value": "41a96952-ab9f-4abe-a2c8-e096c63324db"
    },
    {
      "area_uuid_name": "areas_mashery_internal_area_uuid",
      "area_uuid_value": "23b1c970-dcb5-4fe5-8e01-1a1fcf957cc4"
    },
    {
      "area_uuid_name": "areas_mashery_area_uuid",
      "area_uuid_value": "2014f1e7-1dd6-496b-b9bf-1c4b58bf7332"
    },
    {
      "area_uuid_name": "areas_solutions_area_uuid",
      "area_uuid_value": "02bcdffc-47d0-4951-b3b1-60c5adc3762f"
    }
  ],
  "package_keys": [
    {
      "package_key_name": "your_api_key_for_v3_api",
      "package_key_value": "7wevdpn289e9v4k6g5ndqu6c"
    },
    {
      "package_key_name": "package_key_mapi_222ux5v3yjdj6qqptk2y733xa_value",
      "package_key_value": "ajajruzw7hrpaguddnt3r2mq"
    },
    {
      "package_key_name": "package_key_mapi_4ayzfcn742zspfm6jv48htv_value",
      "package_key_value": "87dgm9vxgwapynf68xk2cjc"
    },
    {
      "package_key_name": "package_key_mapi_4ngdrf2tg62gvgsq7z9fd2f7_value",
      "package_key_value": "2z33rwdddxhndhz7csu566bk"
    },
    {
      "package_key_name": "package_key_mapi_gauhqskrkf4qzuv92esqht6a_value",
      "package_key_value": "d2779hcqjsw2fda445bsr36z"
    }
  ]
}
```

```

    },
    {
      "package_key_name": "package_key_mapi_gtgkmtxq8nnz9ajy86xtr9z4e_value",
      "package_key_value": "ddk3c6btngm39vgckqd3kq4t"
    },
    {
      "package_key_name": "package_key_mapi_qqky3473rqptpsrsvjdc6ah_value",
      "package_key_value": "kyjyn5ystz3f3f7tnep6g79y"
    }
  ],
  "package_secrets": [
    {
      "package_secret_name": "your_secret_for_v3_api",
      "package_secret_value": "nbk27dsyrU"
    },
    {
      "package_secret_name": "package_key_mapi_222ux5v3yjdj6qqptk2y733xa_secret",
      "package_secret_value": "3hw2BKbAfx"
    },
    {
      "package_secret_name": "package_key_mapi_4ayzfcn742zspfmuj6jv48htv_secret",
      "package_secret_value": "kHnrJ6v4xw"
    },
    {
      "package_secret_name": "package_key_mapi_4ngdrf2tg62gvgsq7z9fd2f7_secret",
      "package_secret_value": "jqYwS5twav"
    },
    {
      "package_secret_name": "package_key_mapi_gauhqskrkf4qzuv92esqht6a_secret",
      "package_secret_value": "RkSEpu3N5w"
    },
    {
      "package_secret_name": "package_key_mapi_gtgkmtxq8nnz9ajy86xtr9z4e_secret",
      "package_secret_value": "nXxaYcSCgz"
    },
    {
      "package_secret_name": "package_key_mapi_qqky3473rqptpsrsvjdc6ah_secret",
      "package_secret_value": "Akj8Q645jN"
    }
  ]
}

```

i Note: Do not customize the following properties during initial deployment:

- localdevadmin_handle
- localdevadmin_password
- masherytestautomation_handle

The localdevadmin_password can be changed after initial deployment. To change the default password, see [Changing the Default Admin Password using the Developer Portal](#).

For customization, see *Update Platform API Properties* section in [Deploying the Boomi Cloud™ API Management - Local Edition Cluster](#).

Changing the Default Admin Password using the Developer Portal

The localdevadmin_password property should not be changed from the tml_papi_properties.json file. Even if the password is changed in this file to a different value, the change will not be effective in the running cluster.

To change the default password:

1. Log in to the Developer Portal at https://<CM_EXTERNAL_IP>:8443/; in case of quick start, use <https://localhost:10443/> using the default user name and password.
2. Click on the **User** menu > **Profile**, **Update profile** > **Change password**.
3. Change the password for the logged-in user.
4. **Save** the changes.

Properties per Container Type

The following property files specify property per container type:

```
tml_cache_properties.json
tml_cm_properties.json
tml_log_properties.json
tml_nosql_properties.json
tml_papi_properties.json
tml_sql_properties.json
tml_tm_properties.json
```

tml-tm**tml_tm_properties.json**

Name	Default Value	Comments
auto_binding	ON	When turned ON, sql, cache and logservice URLs will be auto-discovered and bound to the Local Edition component.
http_conn_manager_max_total	256	Caps the maximum supported backend client connections at any given time.
jetty_acceptqueuesize	10 (256 in code)	Jetty Queue size.
jetty_maxidletime	18000 (30000 in code)	Jetty Maximum Idle time.
jetty_pool_maxthreads	500 (500 in code)	Maximum number of jetty threads to be running
logserviceURL	log-set-0-0.log-svc-0.default.svc.cluster.local	Logservice URL is used to communicate with logservice component which processes the Traffic Manager logs.
mail_protocol	smtp	TCP/IP protocol used in sending and receiving e-mail.
mail_smtp_host		The SMTP server to connect to.
mail_smtp_password		Password to connect to SMTP.
mail_smtp_port	25	The SMTP server port to connect to, if the connect() method doesn't explicitly specify one.

Name	Default Value	Comments
mail_smtp_user		Default user name for SMTP.
mail_smtps_host		The SMTPS server to connect to.
mail_smtps_password		Password to connect to SMTPS.
mail_smtps_port	465	The SMTPS server port to connect to, if the connect() method doesn't explicitly specify one.
mail_smtps_user		Default user name for SMTPS.
max_uri_len_for_accesslog	2048	Limits the maximum length of URI data being written to the access log.
java_security_negative_cache_lookups	0	Specified in java.security to indicate the caching policy for unsuccessful name lookups from the name service. The value is specified as an integer to indicate the number of seconds to cache the failure for unsuccessful lookups.
java_security_cache_lookups	0	Specified in java.security to indicate the caching policy for successful name lookups from the name service. The value is specified as an integer to indicate the number of seconds to cache the successful lookup.
nosqlURL	cass-set-0.cass-svc.default.svc.cluster.local	NoSQL URL is used to communicate with registry service.
oauth_http_enabled	TRUE	OAuth HTTP is enabled by default.

Name	Default Value	Comments
oauth_http_port	9083	HTTP port for OAuth.
oauth_https_enabled	TRUE	OAuth HTTPS is enabled by default.
oauth_https_port	8083	HTTPS port for OAuth.
sqlURL	mysql-set-0.mysql-svc.default.svc.cluster.local	SQL URL is used to communicate with the SQL component to get the data.
vmargs	"-d64 \n-Xmx1024m"	VM arguments that typically change the behavior of the Java Virtual Machine (JVM). The Xmx argument allows the Java heap to grow.
use_system_time	true	Use this property for better scalability.

tml-cm

tml_cm_properties.json

Name	Default Value	Comment
auto-binding	ON	When turned ON, logservice URLs will be auto-discovered and bound to the Local Edition cluster component.
domain_notification_from	noreply@mashery.com	
domain_notification_recipient	papi@mashery.com	
http_conn_	1000	Sets the maximum supported

Name	Default Value	Comment
manager_max_total		backend client connections at any given time.
jetty_acceptqueuesize	256	Jetty Queue size
jetty_pool_maxthreads	500	Maximum number of Jetty threads to be running
logserviceURL	log-set-0-0.log-svc-0.default.svc.cluster.local	Logservice URL to connect to for processing the logs.
max_endpoints_per_service_created	600	Maximum endpoints in a service that can be created.
max_services_created	600	Maximum services that can be created.
max_uri_len_for_accesslog	2048	This property is used to limit the maximum length of uri data being written to the access log.
smtp_host		The SMTP server to connect to for portal notifications.
smtp_port	25	The SMTP server port to connect to, if the connect () method doesn't explicitly specify one.
smtp_useTls	False	Indicates if the STARTTLS command be used to initiate a TLS-secured connection to the smtp service
smtp_user		Default user name for SMTP

Name	Default Value	Comment
smtp_password		Password to connect to SMTP
domain_notification_from	noreply@mashery.com	Email address of the account from which email is sent by the system.
domain_notification_recipient	papi@mashery.com	
user_session_validity	300s	Configure the session timeout for the cluster manager

tml-sql

tml_sql_properties.json

Name	Default Value	Comment
api_Key		The API key needed to connect to MOM HOST in Local Edition.
api_Secret		The API password needed to connect to MOM HOST in Local Edition.
auto_binding	ON	When turned ON, logservice URLs will be auto-discovered and bound to the Local Edition component.
data_directory	/var/lib/mysql	The path to the MySQL data directory. Only the last component of the path name is created if it does not exist; the parent directory must already exist or an error occurs.

Name	Default Value	Comment
dimdataLogServiceURL	log-set-0-0.log-svc-0.default.svc.cluster.local	
expire_logs_days	30	Specifies the number of days before automatic removal of binary log files.
innodb_buffer_pool_size	128M	The size in bytes of the buffer pool, the memory area where InnoDB caches table and index data.
key_buffer_size	16M	The size of the buffer used for index blocks. The key buffer is also known as the key cache.
mysql_bin	mysql-bin	Specifies the base name to use for binary log files.
mysql_log_error	/var/log/mysqld.log	Option to determine whether the default error log destination is the console or a file, and, if a file, the file name.
mysql_log_error_verbosity	1	The verbosity for handling events intended for the error log, as filtered by the log_filter_internal error log filter component, which is enabled by default.
logserviceURL	log-set-0.log-svc.default.svc.cluster.local	Logservice URL to connect to process the logs.
max_allowed_packet	16777216	Maximum packet length to send to or receive from server.

Name	Default Value	Comment
max_binlog_size	1073741824	If a write to the binary log causes the current log file size to exceed the value of this variable, the server rotates the binary logs (closes the current file and opens the next one). The minimum value is 4096 bytes. The maximum and default value is 1GB.
mom_host	https://api-mom.mashery.com	MOM host to connect to sync data from Local Edition.
mom_connect_timeout	30000	Sets a specified timeout value, in milliseconds, to be used when opening a communications link to MOM referenced by a URLConnection.
mom_read_timeout	120000	Sets the read timeout to MOM URL to a specified timeout, in milliseconds. A non-zero value specifies the timeout when reading from Input stream when a connection is established to MOM. If the timeout expires before there is data available for read, a <code>java.net.SocketTimeoutException</code> is raised.
plugin_load	validate_password.so	This option tells the server to load the named plugins at startup. If multiple <code>--plugin-load</code> options are given, only the last one is used.

Name	Default Value	Comment
query_cache_size	8M	This system variable was removed in MySQL 8.0.3.
server_id	1234567890	Specifies the server ID. This variable is set by the <code>--server-id</code> option. The <code>server_id</code> system variable is set to 1 by default. The server can be started with this default ID, but when binary logging is enabled, an informational message is issued if you did not specify a server ID explicitly using the <code>--server-id</code> option.
service_sync_interval_minutes	15	Service sync interval to sync service configurations with MOM.
developer_sync_interval_minutes	15	Developer sync interval to sync developer configurations with MOM.
table_open_cache	1000	The number of open tables for all threads. Increasing this value increases the number of file descriptors that mysqld requires.

tml-cache

tml_cache_properties.json

Name	Default Value	Comments
auto_binding	ON	When turned ON, SQL and logservice URLs are auto-discovered and

Name	Default Value	Comments
		bound to the Local Edition component.
content_cachesize	384	Cache size for memcached Content Pool
content_maxconn	10240	Max connection for memcached Content Pool
content_options	-o modern	Options for memcached Content Pool
content_port	11213	Port for memcached Content Pool
content_user	nobody	User for memcached Content Pool
count_cachesize	256	Cache size for memcached Count Pool
count_maxconn	10240	Max connection for memcached Count Pool
count_options	-o modern	Options for memcached Count Pool
count_port	11212	Port for memcached Count Pool
count_user	nobody	User for memcached Count Pool memcache
logserviceURL	log-set-0.log-svc.default.svc.cluster.local	Logservice URL to connect to process the logs.

Name	Default Value	Comments
mem_cachesize	384	Cache size for memcached Pool
mem_maxconn	10240	Max connection for memcached Pool
mem_options	-o modern	Options for memcached Pool
mem_port	11211	Port for memcached Pool
mem_user	nobody	User for memcached Pool
packager_cachesize	128	Cache size for memcached Packager Pool
packager_maxconn	10240	Max connection for memcached Packager Pool
packager_options	-o modern	Options for memcached Packager Pool
packager_port	11215	Port for memcached Packager Pool
packager_user	nobody	User for memcached Packager Pool
services_cachesize	128	Cache size for memcached Service Pool
services_maxconn	10240	Max connection for memcached Service Pool
services_options	-o modern	Options for memcached

Name	Default Value	Comments
		Service Pool
services_port	11214	Port for memcached Service Pool
services_user	nobody	User for memcached Service Pool
sqlURL	mysql-set-0.mysql-svc.default.svc.cluster.local	MYSQL URL to connect to load data from
updateddevclass	10	Sync developer classes updated in the last X minutes. X is the value of this property.
updatedhttpsclientsecurity	30	Sync https client security information updated in the last X minutes. X is the value of this property.
updatedmapi	10	Sync mapis updated in the last X minutes. X is the value of this property.
updatedpackager	30	Sync packagers updated in the last X minutes. X is the value of this property.
updateservice	10	Sync services updated in the last X minutes. X is the value of this property.

tml-log

tml_log_properties.json

Name	Default Value	Comments
api_key		Only needed for "Tethered" setup.
api_secret		Only needed for "Tethered" setup.
td_agent_output_channelType	DEFAULT (always enabled)	Set this property if you need to send access logs to other destinations. Accepted values are DEFAULT, ELASTICSEARCH, KAFKA, HTTP and TCP. Changing to anything other than DEFAULT requires additional properties: Property Name / Default Value • td_agent_out_elasticsearch_host / NULL • td_agent_out_elasticsearch_port / NULL • td_agent_out_elasticsearch_index / \${TAG} • td_agent_out_http_URI / NULL • td_agent_out_kafka_brokers / NULL • td_agent_out_kafka_topic / \${TAG} • td_agent_out_tcp_host / NULL • td_agent_out_tcp_port / NULL
mom_host	https://api-mom.mashery.com	Only needed for "Tethered" setup.

tml-nosql

tml_nosql_properties.json

Name	Default Value	Comment
auto_binding	ON	When turned ON, logservice URLs are auto-discovered and bound to the Local Edition component.
logserviceURL	log-set-0.log-svc.default.svc.cluster.local	Logservice URL to connect to process the logs.

Ports to Open Externally

The following ports

Port Number	Associated Component
8080	Configuration Manager UI
8443	Configuration Manager UI - HTTPS
7080	V3 API
7443	V3 API - HTTPS
80	Traffic Manager - HTTP Traffic
443	Traffic Manager - HTTPS Traffic
1443	Traffic Manager - MTLS
8083	Traffic Manager - OAuth Service
3000	Reporting Service

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 Local Edition cluster in a Kubernetes environment, the cluster was created using the MySQL password specified in the `tml_cluster_properties.json` file.



Note:

There is downtime associated with changing the MySQL password, so it is recommended that any changes be made during a maintenance period.

To change the MySQL password:

Procedure

1. Encrypt the password using the password utility shipped with the installer :

`docker exec -it <installer container id> password_utility.`

```
docker exec -it 7f3a82143625 password_utility
Enter password :
Re-enter password :
Y2hhbmdlbWUxMg==
```

2. After encrypting the password, change the password in the `tml_cluster_properties.json` file in the deployment folder.

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

In this example, modify the below file by updating the generated encoded password as `Y2hhbmdlbWUxMg==`

```
{
  "mysql_root_pwd": "Y2hhbmdlbWUxMg==",
  "mysql_masheryonprem_pwd": "Y2hhbmdlbWUxMg==",
  "mysql_mashonpremrepl_pwd": "Y2hhbmdlbWUxMg==",
  "mysql_mashclient_pwd": "Y2hhbmdlbWUxMg==",
  "mysql_masherybackup_pwd": "Y2hhbmdlbWUxMg==",
  "mom_server": "https://api-mom.mashery.com",
```

```
"mom_key": "",
"mom_secret": ""
}
```

3. Delete the cluster-property Kubernetes secret:

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

4. 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
```

5. Log in to the MySQL container:

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

6. 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';
```

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

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

9. 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
```

10. 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
```

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

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

12. Once all the pods are running, you can log-in 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.

API Debug Key Common to All Deployments

Customizing api_debug_key and api_debug_secret

You can customize the following properties in `docker-deploy/properties/tml_tm_properties.json` file before deploying the Boomi Cloud™ API Management - Local Edition cluster.

```
"api_debug_key": "24NumbersAndOrCharacters",
"api_debug_secret": "10NumChars"
```



Note: The `api_debug_key` is composed of 24 numbers and / or lower-case characters.



Note: The `api_debug_secret` is composed of 10 numbers and / or lower-case characters.

Generating API Debug Authentication Code

You can generate API debug authentication code using the following scripts:

```
php -r 'echo hash_hmac("sha256", "$api_debug_key", "$api_debug_secret");'; echo
```

that is,

```
php -r 'echo hash_hmac("sha256", "24NumbersAndOrCharacters", "10NumChars");'; echo
```

The following is the sample output for reference:

```
398c80380ad5cd32c06e8018108528a6119e33f99a639dfc965bb461392baf02
```

Using API Debug Authentication Code in Requests

You can use debug authentication codes in requests.

To enable API debug, use X-Mashery-Debug as the HTTP header. The value of X-Mashery-Debug can be obtained from the following sample code:

```
curl -v -H 'host: tml.mashery.com' -H 'X-Mashery-Debug:
398c80380ad5cd32c06e8018108528a6119e33f99a639dfc965bb461392baf02'
http://$LB:80/mock?api_key=3rrgjz32q64ysm4t8ea4jzqa
```

An example of request and response is,

```
curl -v -H 'host: tml.mashery.com' -H 'X-Mashery-Debug:
398c80380ad5cd32c06e8018108528a6119e33f99a639dfc965bb461392baf02'
http://$LB:80/mock?api_key=3rrgjz32q64ysm4t8ea4jzqa
* Trying 35.222.3.8...
* TCP_NODELAY set
* Connected to 35.222.3.8 (35.222.3.8) port 80 (#0)
> GET /mock?api_key=3rrgjz32q64ysm4t8ea4jzqa HTTP/1.1
> host: tml.mashery.com
> User-Agent: curl/7.54.0
> Accept: */*
> X-Mashery-Debug: fd4778acb14aa2470a3f1cf7a6d77a2023d319f82cae270a545be66cfc7ac3d9
>
< HTTP/1.1 200 OK
< X-Mashery-Debug-path_info: /mock
```



```

< X-Mashery-Responder: tm-deploy-0-54bcd5489f-7qdkv
< Content-type: text/html
< Date: Sat, 26 Oct 2019 22:12:32 GMT
< X-Powered-By: PHP/5.3.3
< Content-Length: 25
< X-Mashery-Debug-Servlet-Bundle-Version-com.mashery.proxy.servlet: 1.0.0.v20191026-1954
< X-Mashery-Debug-Responder-Bundle-Version-com.mashery.proxy.responder: 1.0.0.v20190812-0255
< X-Mashery-Debug-Request-Method: GET
< X-Mashery-Debug-Endpoint-Sorted: [/mock]
< X-Mashery-Debug-Endpoint-Match-Checking0: /mock
< X-Mashery-Debug-Endpoint-Match: /mock
< X-Mashery-Debug-Key: GET tml.mashery.com/mock
< X-Mashery-Debug-Service-Key: 6t5mcv6aap3xzsyaf3qd5b4p::171230
< X-Mashery-Debug-Service-Revision-Number: 2
< X-Mashery-Debug-Adapter-Factory: com.mashery.proxy.responder.DefaultAdapter
< X-Mashery-Debug-Adapter-Bundle-Version-com.mashery.proxy.responder: 1.0.0.v20190812-0255
< X-Mashery-Debug-Split-Path-Method-Found:
< X-Mashery-Debug-Find-Mapi-Value-Data: [api_key=3rrgjz32q64ysm4t8ea4jzqa]
< X-Mashery-Debug-Mapi-Value-Key: api_key
< X-Mashery-Debug-PLAN-KEY: Plan_6310::171230::-1
< X-Mashery-Debug-PLAN-IS_UNDEFINED_METHOD_ALLOWED: true
< X-Mashery-Debug-Memcount-Timestamp: 1572127951
< X-Mashery-Debug-Plan-QPS-Limit: 2
< X-Mashery-Debug-QPS-Dev-Key: QPS_6310::PKG::3rrgjz32q64ysm4t8ea4jzqa::pid6310:31
< X-Mashery-Debug-Rate-Count-Component-Name: pid
< X-Mashery-Debug-Rate-Count: 2
< X-Mashery-Debug-Rate-Limit-Ceiling: 5000
< X-Mashery-Debug-Cache-Enabled: 0
< X-Mashery-Debug-Backend-Request: http://content.mock.mashery.com/?api
key=3rrgjz32q64ysm4t8ea4jzqa
< X-Mashery-Debug-Backend-Addresses-All: [content.mock.mashery.com/50.17.201.93,
content.mock.mashery.com/54.204.22.178,
content.mock.mashery.com/2406:da00:ff00:0:0:0:3211:c95d,
content.mock.mashery.com/2406:da00:ff00:0:0:0:36cc:16b2]
< X-Mashery-Debug-Backend-Connect: content.mock.mashery.com/54.204.22.178:80
< X-Mashery-Debug-Backend-Address: content.mock.mashery.com/54.204.22.178:80
< X-Mashery-Debug-Default-Do-Filter: false
<
* Connection #0 to host 35.222.3.8 left intact

```

Deployment Sequence

The following section provides information on the sequence to be followed for deploying a Local Edition cluster using custom scripts or any combined practices of Continuous Integration (CI) and Continuous Delivery (CD).

**Note:**

- Local Edition depends on a proper sequence due to dependencies between each of the pods. Each pod also advertises its state with the registry. Because of this registry requirement the NoSQL pods must be deployed before any other pod.
- In case deployment scripts provided by Boomi are used, the following deployment sequence can be ignored.

Before you begin

- The service pods must be created in the following order:
 1. NoSQL
 2. Configuration Manager
 3. Log
 4. MySQL
 5. Cache
 6. Traffic Manager
 7. Reporting
- Run `compose.sh` with the topology specific manifest file and generate the deployment scripts.
- Depending on the deployment topology, the deployment scripts are located at the following locations:
 1. Single zone : "<tml-base>/docker-deploy/kubernetes/manifest"
 2. Multi zone: "<tml-base>/docker-deploy/kubernetes/manifest"
- For multi-zone deployment for tml-type apply resources start from index 0, then 1 and so on for every zone.

i Note: For OpenShift replace `kubectl` with `oc`.

Procedure

1. Create resource secrets.

```
kubectl create secret generic nosql-resource --from-file=resources/tml-nosql
kubectl create secret generic cm-resource --from-file=resources/tml-cm
kubectl create secret generic log-resource --from-file=resources/tml-log
kubectl create secret generic sql-resource --from-file=resources/tml-sql
kubectl create secret generic cache-resource --from-file=resources/tml-cache
kubectl create secret generic tm-resource --from-file=resources/tml-tm
```

2. Create secret as follows:

```
kubectl create secret generic cluster-property --from-file=tml_cluster_properties.json
kubectl create secret generic zones-property --from-file=tml_zones_properties.json
kubectl create secret generic papi-property --from-file=tml_papi_properties.json
kubectl create secret generic nosql-property --from-file=tml_nosql_properties.json
kubectl create secret generic cm-property --from-file=tml_cm_properties.json
kubectl create secret generic cm-jks --from-file=tml-cm.jks
kubectl create secret generic cm-crt --from-file=tml-cm-crt.pem
kubectl create secret generic cm-key --from-file=tml-cm-key.pk8
kubectl create secret generic tm-property --from-file=tml_tm_properties.json
kubectl create secret generic tm-jks --from-file=tml-tm.jks
kubectl create secret generic tm-trust-jks --from-file=tml-tm-trust.jks
kubectl create secret generic sql-property --from-file=tml_sql_properties.json
kubectl create secret generic log-property --from-file=tml_log_properties.json
kubectl create secret generic cache-property --from-file=tml_cache_properties.json
```

For deploying tml-reporting create the following secrets.

```
kubectl create secret generic reporting-resource --from-file=resources/tml-reporting
kubectl create secret generic fluentd-conf --from-file=resources/tml-reporting/fluentd/conf
kubectl create secret generic fluentd-plugin --from-file=resources/tml-reporting/fluentd/plugin
kubectl create secret generic grafana-resource --from-file=resources/tml-reporting/grafana
kubectl create secret generic grafana-user-dashboards --from-file=resources/tml-reporting/grafana/dashboards/CustomDashboards
kubectl create secret generic grafana-operations --from-file=resources/tml-reporting/grafana/dashboards/MasheryReporting/operations
kubectl create secret generic grafana-summary --from-file=resources/tml-reporting/grafana/dashboards/MasheryReporting/summary
kubectl create secret generic prometheus-resource --from-file=resources/tml-
```

```
reporting/prometheus
kubectl create secret generic loki-resource --from-file=resources/tml-reporting/loki
```

3. If you want to create an image pull secret.

```
kubectl create secret docker-registry \
  --docker-server=<registry host> \
  --docker-username=<registry username> \
  --docker-password=<registry password> \
  tmgc-registry-key
```

4. Deploy all services as follows:

```
kubectl apply -f cass-svc- $\{index\}$ .yaml
kubectl apply -f log-svc- $\{index\}$ .yaml
kubectl apply -f mysql-svc- $\{index\}$ .yaml
kubectl apply -f cache-svc- $\{index\}$ .yaml
kubectl apply -f cm-svc- $\{index\}$ .yaml
kubectl apply -f tm-svc.yaml
```



Note: Reporting service can be deployed only in a single zone.

Sample commands for us-central1-a zone is as follows:

```
kubectl apply -f reporting-svc-us-central1-a.yaml
kubectl apply -f reporting-svc-external-us-central1-a.yaml
```

5. Check if the services have been successfully created as follows:

```
kubectl get services
```

Output will be as follows:

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP
cache-svc-0	ClusterIP	None	<none>
11211/TCP,11212/TCP,11213/TCP,11214/TCP,11215/TCP,11216/TCP			9d
cass-svc-0	ClusterIP	None	<none>
			9042/TCP

```

          9d
cm-svc-0    LoadBalancer 10.100.59.255 aa5a3174ced1f4813b1ee93c4d811f6b-
1898460514.us-east-1.elb.amazonaws.com
8443:31577/TCP,7080:31954/TCP,7443:31677/TCP          9d
kubernetes  ClusterIP    10.100.0.1    <none>
443/TCP                                           10d
log-svc-0   ClusterIP    None        <none>
24224/TCP,24220/TCP,24221/TCP,24222/TCP          9d
mysql-svc-0 ClusterIP    None        <none>
3306/TCP,33061/TCP                               9d
reporting-app-0 LoadBalancer 10.100.175.69 a6987a86fa3ff4bfc8f27073ee2217db-
774902445.us-east-1.elb.amazonaws.com 3000:32563/TCP 7d
reporting-svc-0 ClusterIP    None        <none>
3000/TCP                                           9d
tm-svc      LoadBalancer 10.100.174.66 a010ad409036a45a2ab99782f4a8706f-
169206996.us-east-1.elb.amazonaws.com
80:32479/TCP,443:31313/TCP,8083:31806/TCP          9d
tm-svc-internal LoadBalancer 10.100.54.230 a830c47aea66f4ac1be79c3875d51f21-
1622402344.us-east-1.elb.amazonaws.com
80:30371/TCP,443:30671/TCP,8083:31015/TCP          9d

```

6. Deploy NoSQL service.

```
kubectl apply -f nosql-pod-${index}.yaml
```

Note: After applying NoSQL pod you will see the following log output indicating Cassandra nodes are up and running.

```
kubectl logs cass-set-0-0
```

Output will be as follows:

```

SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in
[jar:file:/opt/mashery/containeragent/lib/tpcl/ch.qos.logback.classic_
1.2.3.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/opt/mashery/containeragent/lib/tpcl/logback-classic-
1.2.3.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.

```

```
INFO [main] 2021-08-07 01:17:01,138 Gossiper.java:1701 - No gossip backlog; proceeding
SLF4J: Actual binding is of type [ch.qos.logback.classic.util.ContextSelectorStaticBinder]
containeragent (pid 1276) is running...
```

If there are multiple Cassandra pods, the below log output can be seen.

```
INFO [main] 2021-07-30 07:47:23,503 StorageService.java:1449 - JOINING: Starting to
bootstrap...
INFO [main] 2021-07-30 07:47:23,545 StreamResultFuture.java:90 - [Stream #60732d20-
f10a-11eb-9dee-6fb0c9b1b7b6] Executing streaming plan for Bootstrap
INFO [StreamConnectionEstablisher:1] 2021-07-30 07:47:23,554 StreamSession.java:266 -
[Stream #60732d20-f10a-11eb-9dee-6fb0c9b1b7b6] Starting streaming to /192.168.8.107
INFO [StreamConnectionEstablisher:1] 2021-07-30 07:47:23,557
StreamCoordinator.java:264 - [Stream #60732d20-f10a-11eb-9dee-6fb0c9b1b7b6, ID#0]
Beginning stream session with /192.168.8.107
INFO [STREAM-IN-/192.168.8.107:7000] 2021-07-30 07:47:23,568
StreamResultFuture.java:187 - [Stream #60732d20-f10a-11eb-9dee-6fb0c9b1b7b6]
Session with /192.168.8.107 is complete
INFO [StreamConnectionEstablisher:2] 2021-07-30 07:47:23,575 StreamSession.java:266 -
[Stream #60732d20-f10a-11eb-9dee-6fb0c9b1b7b6] Starting streaming to /192.168.19.160
INFO [StreamConnectionEstablisher:2] 2021-07-30 07:47:23,580
StreamCoordinator.java:264 - [Stream #60732d20-f10a-11eb-9dee-6fb0c9b1b7b6, ID#0]
Beginning stream session with /192.168.19.160
INFO [STREAM-IN-/192.168.19.160:7000] 2021-07-30 07:47:23,588
StreamResultFuture.java:187 - [Stream #60732d20-f10a-11eb-9dee-6fb0c9b1b7b6]
Session with /192.168.19.160 is complete
INFO [STREAM-IN-/192.168.19.160:7000] 2021-07-30 07:47:23,593
StreamResultFuture.java:219 - [Stream #60732d20-f10a-11eb-9dee-6fb0c9b1b7b6] All
sessions completed
INFO [main] 2021-07-30 07:47:23,599 StorageService.java:1449 - JOINING: Finish joining
ring
INFO [STREAM-IN-/192.168.19.160:7000] 2021-07-30 07:47:23,606
StorageService.java:1508 - Bootstrap completed! for the tokens [7635535438842737552,
6244423451386580644, -807892735552018602, 17969032
38882583306, -959993705797373391, -4431754111306089197, -7456837306080462267, -
5393060984937455814, 7879194275152297156, -6332960200985977913, -
566523580936125161, -2328256701323223797, 353452392273
0543018, 3768424131407991242, -2080329565188323460, -1951378018385099350,
4110905109422783803, -1123641547224386520, -957528312907090424, -
31625905873932128, 6347044696158661417, 2642191943803334943
, -1292908263869730037, -6202545713550248129, 2028717097417634416,
2283340219657285471, 5821394578554330785, 6084908708426866506,
```

```
5343046259693124960, -4159072616317009065, 6907637449959885635, -594
1635556830205750]
```

7. Deploy the Configuration Manager.

```
kubectl apply -f cm-pod-${index}.yaml
```

8. Wait for Cassandra SQL database to get ready and then check the registry log.

```
[2021-08-09T06:40:00,889Z] RegistryService [QuartzScheduler_Worker-2] INFO
com.tibco.mashery.tmgc.registry.cron.CassandraStatusCheckerJob - total nodes 3, healthy
nodes 3, percentage 100%
[2021-08-09T06:40:00,889Z] RegistryService [QuartzScheduler_Worker-2] INFO
com.tibco.mashery.tmgc.registry.cron.CassandraStatusCheckerJob - Cassandra is healthy
and Registry keyspaces available : true
[2021-08-09T06:40:30,008Z] RegistryService [QuartzScheduler_Worker-3] INFO
com.tibco.mashery.tmgc.registry.cron.OrphanTmgcCheckerJob - Orphan check completed
successfully in the clusterId 6753d591-e0d1-4b6d-a96f-cd4b169bf1e8 and zoneId d8dffe8d-
5f27-45f8-8162-6d1c6783bf8f
[2021-08-09T06:42:00,864Z] RegistryService [QuartzScheduler_Worker-2] INFO
com.tibco.mashery.tmgc.registry.cron.CassandraStatusCheckerJob - nodetool status
Datacenter: dc1
=====
Status=Up/Down
|/ State=Normal/Leaving/Joining/Moving
-- Address      Load      Tokens    Owns (effective)  Host ID                               Rack
UN 192.168.6.8   3.06 MiB  32        100.0%           588018d1-0fe6-4e7d-9f51-
67661048822d rack1
UN 192.168.30.112 2.97 MiB  32        100.0%           bd0f3a65-791b-49bd-bc50-
23b6ef92b97a rack1
UN 192.168.18.112 3.05 MiB  32        100.0%           61f63784-573d-4772-ae46-
cc3cea5d3ea1 rack1
```

9. Deploy the Log pod.

```
kubectl apply -f log-pod-${index}.yaml
```

10. Deploy the MySQL pod.

```
kubectl apply -f sql-pod-${index}.yaml
```

11. Deploy the Cache pod.

```
kubectl apply -f cache-pod-${index}.yaml
```

12. Deploy the Traffic Manager pod.

```
kubectl apply -f tm-pod-${index}.yaml
```

13. Optional: If you want to deploy reporting services carry out the following steps.

- Label one of the node as `kubectl label node <node> node-name=reporting`.



Note: For multi-zones, the node must start from the first zone.

- Deploy reporting pods.

```
kubectl apply -f reporting-pod-0.yaml
```

Result

Check the status of all the pods.

```
kubectl get pods
```

Output

NAME	READY	STATUS	RESTARTS	AGE
cache-set-0-0	1/1	Running	0	2d5h
cache-set-0-1	1/1	Running	0	2d5h
cass-set-0-0	1/1	Running	0	2d5h
cass-set-0-1	1/1	Running	0	2d5h
cass-set-0-2	1/1	Running	0	2d5h
cm-deploy-0-7b6b96bbf5-sbxmd	1/1	Running	0	2d5h
log-set-0-0	1/1	Running	0	2d5h
log-set-0-1	1/1	Running	0	2d5h
mysql-set-0-0	1/1	Running	0	2d5h
reporting-set-0-0	1/1	Running	0	2d5h
tm-deploy-0-64fb5f79b4-qsjc4	1/1	Running	0	2d5h
tm-deploy-0-64fb5f79b4-rhvzn	1/1	Running	0	2d5h
tm-deploy-0-64fb5f79b4-s6cr4	1/1	Running	0	2d5h

MySQL Replication Groups

Local Edition cluster in untethered mode uses MySQL replication groups.

In the case of one of the zones going down in a multi zone cluster, to achieve high availability, three or more zones are required.

If there are three or more zones, one or more tml-sql pods in each zone are needed to ensure MySQL's availability.

 **Note:** Quorum = .Number of members / 2 + 1

To achieve true high availability, a minimum number of 3 zones is required; two or more tml-sql pods in one zone are better than single tml-sql pod in one zone since loads are distributed among available tml-sql pods.

The following section covers MySQL replication groups, and how to rebuild one for an untethered Kubernetes-based Local Edition cluster.

- [Normal State of MySQL Replication Group](#)
- [Abnormal State of MySQL Replication Group](#)
- [Rebuilding MySQL Replication Group](#)

Normal State of MySQL Replication Group

All members of MySQL replication group should be ONLINE.

The following example shows a MySQL replication group with three members:

```
mysql-set-0-0.mysql-svc-2.default.svc.cluster.local
mysql-set-1-0.mysql-svc-0.default.svc.cluster.local
mysql-set-2-0.mysql-svc-1.default.svc.cluster.local
```

Running the following commands should yield the same results:

```
kubectl exec mysql-set-0-0 -- mysql -uroot -pchangeme -e "SELECT * FROM performance_
schema.replication_group_members;"
kubectl exec mysql-set-1-0 -- mysql -uroot -pchangeme -e "SELECT * FROM performance_
schema.replication_group_members;"
kubectl exec mysql-set-2-0 -- mysql -uroot -pchangeme -e "SELECT * FROM performance_
schema.replication_group_members;"
```

Example output:

```
CHANNEL_NAME  MEMBER_ID  MEMBER_HOST  MEMBER_PORT  MEMBER_STATE
group_replication_applier  070f57ea-e271-11ea-bca7-ce95fa60e6f7  mysql-set-2-0.mysql-svc-
2.default.svc.cluster.local 3306  ONLINE
group_replication_applier  836c7441-e270-11ea-b6fa-4ea77323e96f  mysql-set-0-0.mysql-svc-
0.default.svc.cluster.local 3306  ONLINE
group_replication_applier  c7ae2e67-e270-11ea-ba2b-ea4a486569d2  mysql-set-1-0.mysql-svc-
1.default.svc.cluster.local 3306  ONLINE
```

The above output show all three members are online.

Abnormal State of MySQL Replication Group

When some members fail, the number of remaining members is less than quorum, and the MySQL replication group is compromised.

For example, run the following commands to delete two members:

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

The two members "mysql-set-1-0" and "mysql-set-2-0" are deleted.

Since they are in Kubernetes stateful set, they will be recreated by Kubernetes. Before the two members are recreated, the original MySQL replication group is compromised and cannot recover without manual intervention.

The followings are example outputs from each member after the two members are recreated:

```
ubectl exec mysql-set-0-0 -- mysql -uroot -pchangeme -e "SELECT * FROM performance_
schema.replication_group_members;"
```

```
CHANNEL_NAME  MEMBER_ID  MEMBER_HOST  MEMBER_PORT  MEMBER_STATE
group_replication_applier  070f57ea-e271-11ea-bca7-ce95fa60e6f7  mysql-set-2-0.mysql-svc-
2.default.svc.cluster.local 3306  UNREACHABLE
group_replication_applier  836c7441-e270-11ea-b6fa-4ea77323e96f  mysql-set-0-0.mysql-svc-
0.default.svc.cluster.local 3306  ONLINE
group_replication_applier  c7ae2e67-e270-11ea-ba2b-ea4a486569d2  mysql-set-1-0.mysql-svc-
1.default.svc.cluster.local 3306  UNREACHABLE
```

```
kubectl exec mysql-set-1-0 -- mysql -uroot -pchangeme -e "SELECT * FROM performance_
schema.replication_group_members;"
```

```
CHANNEL_NAME  MEMBER_ID  MEMBER_HOST  MEMBER_PORT  MEMBER_STATE
group_replication_applier  c7ae2e67-e270-11ea-ba2b-ea4a486569d2  mysql-set-1-0.mysql-svc-
1.default.svc.cluster.local 3306  ONLINE
```

```
kubectl exec mysql-set-2-0 -- mysql -uroot -pchangeme -e "SELECT * FROM performance_
schema.replication_group_members;"
```

```
CHANNEL_NAME  MEMBER_ID  MEMBER_HOST  MEMBER_PORT  MEMBER_STATE
group_replication_applier  070f57ea-e271-11ea-bca7-ce95fa60e6f7  mysql-set-2-0.mysql-svc-
2.default.svc.cluster.local 3306  ONLINE
```

Rebuilding MySQL Replication Group

When the MySQL replication group is compromised:

1. Stop using any group member.
2. Rebuild the replication group.

Additional actions are described below.

Backup Data in MySQL Database

Login to the last used member (the member in the zone in which admin uses Cluster Manager), run the following command to backup data in MySQL database. For example:

```
mysqldump -uroot -pchangeme --set-gtid-purged=OFF --no-create-info --skip-triggers --ignore-
table=masherysolar.audit_agents --ignore-table=masherysolar.migration_log masherysolar >
/tmp/masherysolar.sql
```

Run the following command to copy backup data to admin workstation. For example:

```
kubectl cp mysql-set-1-0:/tmp/masherysolar.sql ~/tmp/masherysolar.sql
```

i Note: To find the last used member, run the following command. This example lists all services:

```
kubectl get svc
```

The following example output shows all "cm-svc":

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)
cm-svc-0	LoadBalancer	10.xxx.xx.34	35.xxx.xxx.45	8080:30908/TCP,8443:30291/TCP,7080:30820/TCP,7443:31435/TCP 29m
cm-svc-1	LoadBalancer	10.xxx.x.118	35.xxx.xxx.108	8080:32030/TCP,8443:31889/TCP,7080:30467/TCP,7443:30496/TCP 29m
cm-svc-2	LoadBalancer	10.xxx.x.100	35.xxx.xx.47	8080:31734/TCP,8443:31175/TCP,7080:32285/TCP,7443:30027/TCP 29m

If admin access cm-svc at "35.xxx.xxx.45", admin uses "cm-svc-0". In this case, the tml-sql pod is "mysql-set-0-0".

If mysqldump is stuck in last used member, make a backup of data in other members.

Undeploy All MySQL Members

Run the following command to undeploy all MySQL members:

```
./undeploy-sql-pod.sh
```

Delete Persistent Volumes for MySQL Members

Run the following command to delete persistent volumes for MySQL members:

```
kubectl delete pvc $(kubectl get pvc -o=jsonpath='{.items[?(@.metadata.labels.app=="mysql-svc")].metadata.name}')
```

Update Deployment Manifest

Update the deployment manifest to put the deployment in MySQL replication group rebuilding state.

Find the following line in the deployment manifest:

```
"tml_sql_restore": "false",
```

Change this line to:

```
"tml_sql_restore": "true",
```

Generate updated deployment scripts, for example:

```
./compose manifest.yaml
```

Deploy All MySQL Members

To deploy all MySQL members, run the following command:

```
./deploy-sql-pod.sh
```



Note: If the deployment of MySQL members is not successful, repeat the following steps:

1. Undeploy all MySQL members.
2. Delete Persistent Volumes for MySQL members.
3. Deploy all MySQL members.

Copy MySQL Database Backup Data to Any member

To copy MySQL database backup data to any member, run the following command:

```
kubectl cp ~/tmp/masherysolar.sql mysql-set-1-0:/tmp
```

Import Backup Data to MySQL Database

To import backup data, or restore data to MySQL database in the tml-sql pod, run the following command:

```
mysql -uroot -pchangeme masherysolar < /tmp/masherysolar.sql
```

Restart Jetty (OnPremLoader application) process

For **every** tml-sql pod in the cluster, restart the Jetty process that runs the OnPremLoader application. For example, log into **mysql-set-0-0** and then restart Jetty. For example:

```
#log in to the mysql-set-0-0 pod
kubectl exec -it mysql-set-0-0 bash

#once logged into the pod, restart Jetty.
> service jetty restart
```

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

Refer to the [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, 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

Installing MySQL Standard (Commercial) Edition in Local Edition

Local Edition 5.3.1 has a default build configuration that will download and install MySQL Community Edition. For those who prefer to use MySQL Standard (Commercial) edition instead, the build configuration can be updated to do so.

To install MySQL Standard (Commercial) Edition in Local Edition, perform the following steps:

1. Download MySQL Commercial Edition from your subscription.

MySQL commercial edition 5.7.28 is verified for Local Edition 5.3.1. Here is the list of RPMs to download:

- mysql-commercial-libs-5.7.28-1.1.el7.x86_64.rpm
- mysql-commercial-client-5.7.28-1.1.el7.x86_64.rpm
- mysql-commercial-libs-compatible-5.7.28-1.1.el7.x86_64.rpm
- mysql-commercial-common-5.7.28-1.1.el7.x86_64.rpm

- mysql-commercial-server-5.7.28-1.1.el7.x86_64.rpm
2. Copy RPMs of MySQL Commercial Edition to CentOS Linux VirtualBox.
 - a. In CentOS Linux VirtualBox, install package "createrepo":

```
yum install createrepo
```

- b. Create folder "tmgc-sql":

```
mkdir tmgc-sql
```

- c. Copy all RPMs of MySQL Commercial Edition to "tmgc-sql" folder.

3. Prepare RPM Repository.

In CentOS Linux VirtualBox, run the following command in "tmgc-sql" folder:

```
createrepo .
```

4. Copy RPM Repository of MySQL Commercial Edition to Installer host.

In CentOS Linux VirtualBox, run the following command:

```
scp -r tmgc-sql/TML_INSTALLER_HOST_IP:/home/tmlops
```

5. Copy RPM Repository of MySQL Commercial Edition to Installer.

In the host of Installer, start the Installer (For more information, see [Deploying the Installer](#)).

After Installer is started, run the following command:

```
docker cp /home/tmlops/tmgc-sql/ tml-installer:/var/jenkins_home/userContent/tpcl-rpm
docker exec -u root tml-installer chown -R jenkins.jenkins /var/jenkins_
home/userContent/tpcl-rpm/tmgc-sql
```

6. Modify the installation script to install MySQL Commercial Edition.

- a. Login to Installer:

```
docker exec -it tml-installer /bin/bash
```

- b. Modify installation script:


```
vi /var/jenkins_home/docker-build/tmgc-sql/install-mysql/sql-install.sh
```

- c. Uncomment the following lines:

```
cp /home/builder/docker-build/mysql-local.repo /etc/yum.repos.d/
sed -i -e "s/tmgc-installer-host/$HOST_IP/g" /etc/yum.repos.d/mysql-local.repo
sed -i -e "s/tmgc-installer-port/$HOST_PORT/g" /etc/yum.repos.d/mysql-local.repo
```

- d. Comment out the following lines:

```
#cp /home/builder/docker-build/mysql-community.repo /etc/yum.repos.d
#cp /home/builder/docker-build/RPM-GPG-KEY-mysql /etc/pki/rpm-gpg/
```

- e. Comment out the following lines:

```
#yum_install mysql-community-common.x86_64
#yum_install mysql-community-libs.x86_64
#yum_install mysql-community-libs-compat.x86_64
#yum_install mysql-community-client.x86_64
#yum_install mysql-community-server.x86_64
```

- f. Uncomment the following lines:

```
yum_install mysql-commercial-common.x86_64
yum_install mysql-commercial-libs.x86_64
yum_install mysql-commercial-libs-compat.x86_64
yum_install mysql-commercial-client.x86_64
yum_install mysql-commercial-server.x86_64
```

- g. Build Local Edition! Docker images. (For more information, see [Building and Verifying the Docker Images](#).)

The following outputs show MySQL Commercial Edition is installed:

```
---> Package mysql-commercial-common.x86_64 0:5.7.28-1.1.el7 will be installed
---> Package mysql-commercial-libs.x86_64 0:5.7.28-1.1.el7 will be installed
---> Package mysql-commercial-libs-compat.x86_64 0:5.7.28-1.1.el7 will be installed
```

```

---> Package mysql-commercial-client.x86_64 0:5.7.28-1.1.el7 will be installed
---> Package mysql-commercial-server.x86_64 0:5.7.28-1.1.el7 will be installed

```

Deployments on Amazon AWS

Prerequisites

Software Prerequisites

i Note: Ensure you have installed the following required software's and these are available in your PATH before proceeding.

Software	Version Number	Links
Docker	20.x	https://docs.docker.com/engine/install/
AWS CLI	2.7.24	https://docs.aws.amazon.com/cli/latest/userguide/getting-started-install.html
yq	4.27.5	https://github.com/mikefarah/yq/releases/tag/v4.27.5
kubectl	1.23	https://kubernetes.io/docs/tasks/tools/install-kubectl-linux/
eksctl	0.115.0	https://docs.aws.amazon.com/eks/latest/userguide/eksctl.html
oc (only for OpenShift platforms)		https://mirror.openshift.com/pub/openshift-v4/clients/ocp/

Platform Prerequisites

Service Account

To deploy on AWS, a service account **SecretAccessKey** and **AccessKeyID** CSV file is required.

If you do not have a service account already, you can create one by following [these steps](#).

Configuration

To configure the TML Cluster on Amazon AWS, see [Common Configurations for all Deployment Platforms](#)

Configuring AWS Access in TML Installer

Run Jenkins job "configure_aws" on **Prepush** tab to configure the access to AWS in TML Installer.

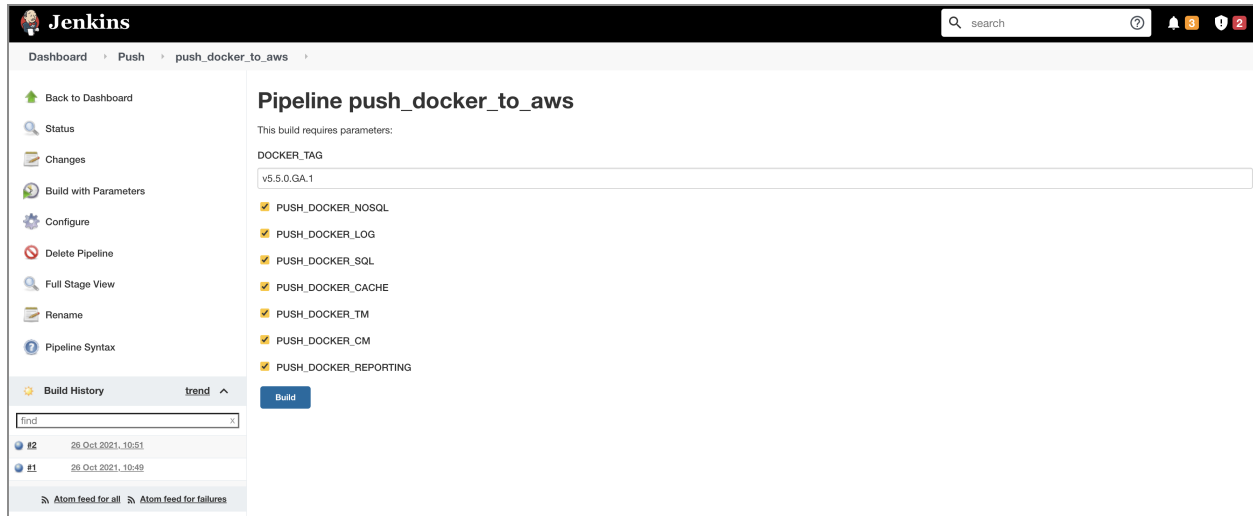
Configure the following required parameters, then click "**Build**"; information will be saved to `"/root/.aws"` in TML Installer.

Parameter	Parameter Value	Description
aws_access_key_id	AWS access key id	Required
aws_secret_access_key	AWS secret access key	Required
aws_default_region	AWS region	This is the region in which AWS Kubernetes cluster and TML cluster will be created.
aws_session_token	AWS assume role	Required, if AWS role-based access control is used.

Push TML Docker Images

Before getting started with this step, ensure the images have been built already. If not, run the Jenkins Job "build_docker".

Run Jenkins job "push_docker_to_aws" on "Push" tab to push TML Docker images to AWS container registry.



Parameter	Description
DOCKER_TAG	the tag for TML Docker images, for example: "v5.6.0.GA.1"
Note: "5.6.0" is TML release; the last "1" is the build number in Jenkins job "build_docker".	
PUSH_DOCKER_NOSQL	Pushes TML Docker image tml-nosql to AWS Container Registry
PUSH_DOCKER_LOG	Pushes TML Docker image tml-log to AWS Container Registry
PUSH_DOCKER_SQL	Pushes TML Docker image tml-sql to AWS Container Registry
PUSH_DOCKER_CACHE	Pushes TML Docker image tml-cache to AWS Container Registry

Parameter	Description
PUSH_DOCKER_TM	Pushes TML Docker image tml-tm to AWS Container Registry
PUSH_DOCKER_CM	Pushes TML Docker image tml-cm to AWS Container Registry
PUSH_DOCKER_API	Pushes TML Docker image tml-api to AWS Container Registry

Deployments on Microsoft Azure

Prerequisites

Software Prerequisites

Note: Ensure you have installed the following required softwares and these are available in your PATH before proceeding.

Software	Version Number	Links
Docker	20.x	https://docs.docker.com/engine/install/
Azure CLI 2.41.0	2.41.0	https://learn.microsoft.com/en-us/cli/azure/install-azure-cli
yq	4.27.5	https://github.com/mikefarah/yq/releases/tag/v4.27.5
kubectl	1.24	https://kubernetes.io/docs/tasks/tools/install-kubectl-linux/
oc (only for OpenShift platforms)		https://mirror.openshift.com/pub/openshift-v4/clients/ocp/

Configuring

To configure the TML Cluster on Microsoft Azure, see [Common Configurations for all Deployment Platforms](#)

Configuring Azure Access in TML Installer

Run Jenkins job `configure_azure` on **Prepush** tab to configure the access to Azure in TML Installer.

The screenshot shows the Jenkins interface for the 'configure_azure' job. The sidebar on the left contains navigation links. The main content area is titled 'Project configure_azure' and lists the following parameters and their values:

- DIRECTORY_ID**: Directory (tenant) ID
- APPLICATION_ID**: Application (client) ID
- CLIENT_SECRET**: Client Secret
- RESOURCE_GROUP**: tmlResourceGroup
- RESOURCE_GROUP_REGION**: eastus
- CONTAINER_REGISTRY**: tmlContainerRegistry

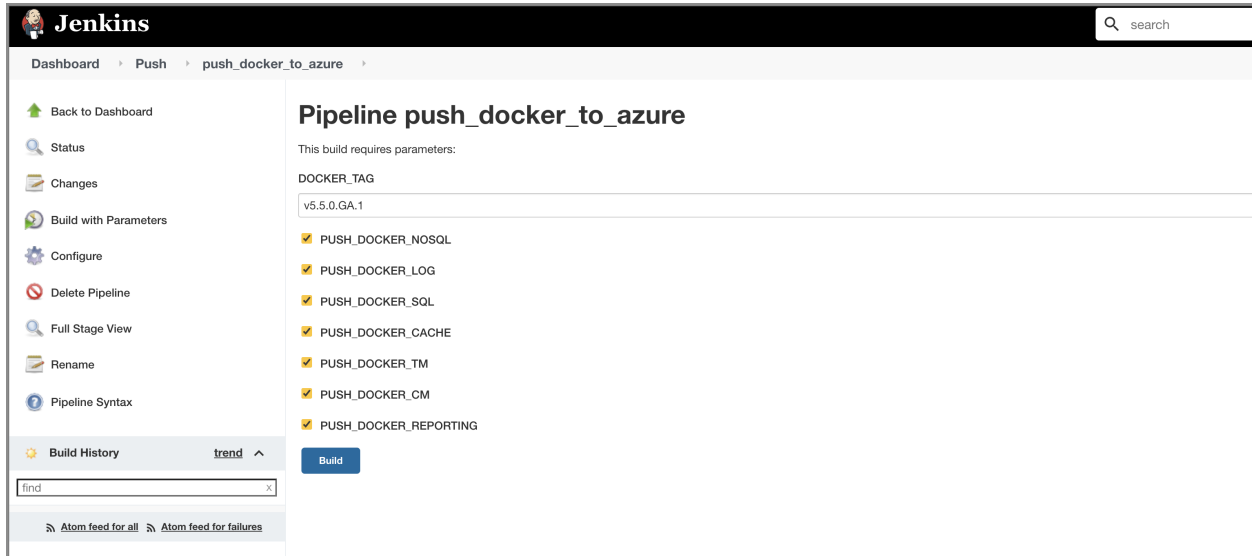
A 'Build' button is located at the bottom right of the parameter list.

Name	Parameter Value	Description
DIRECTORY_ID	Directory (tenant) ID	
APPLICATION_ID	Application (client) ID	Required
CLIENT_SECRET	Client Secret	
RESOURCE_GROUP	Resource Group	The resource group is created if it does not exist.
RESOURCE_GROUP_REGION	Resource Group Region	
CONTAINER_REGISTRY	Container Registry	The container registry is created if it does not exist.

Push TML Docker Images

Before getting started with this step, ensure the images have been built already. If not, run the Jenkins Job "build_docker".

Run Jenkins job "push_docker_to_azure" on "Push" tab to push TML Docker images to Azure container registry.



Parameter	Description
DOCKER_TAG	the tag for TML Docker images, for example: "v5.6.0.GA.1" Note: "5.6.0" is TML release; the last "1" is the build number in Jenkins job "build_docker".
PUSH_DOCKER_NOSQL	Pushes TML Docker image tml-nosql to AWS Container Registry
PUSH_DOCKER_LOG	Pushes TML Docker image tml-log to AWS Container Registry
PUSH_DOCKER_SQL	Pushes TML Docker image tml-sql to AWS Container Registry
PUSH_DOCKER_CACHE	Pushes TML Docker image tml-cache to AWS Container Registry

Parameter	Description
PUSH_DOCKER_TM	Pushes TML Docker image tml-tm to AWS Container Registry
PUSH_DOCKER_CM	Pushes TML Docker image tml-cm to AWS Container Registry

Deployments on Google Cloud Platform

Prerequisites

Software Prerequisites

Note: Ensure you have installed the following required softwares and these are available in your PATH before proceeding.

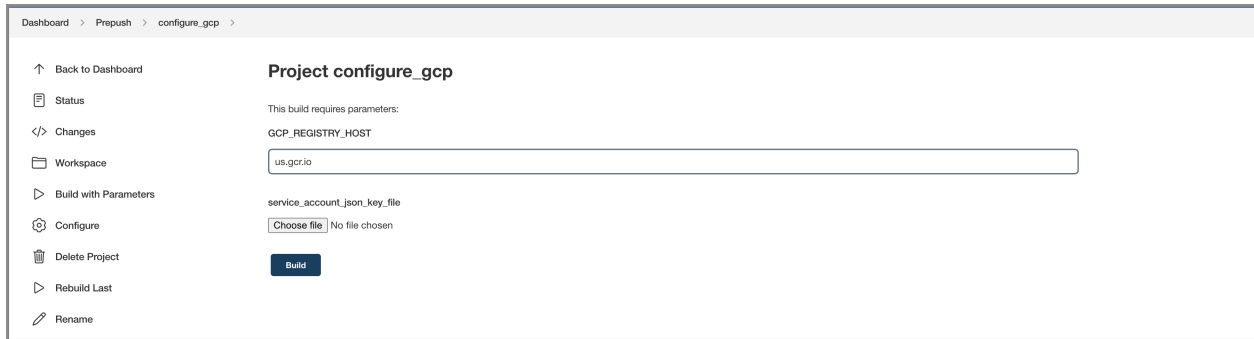
Software	Version Number	Links
Docker	20.x	https://docs.docker.com/engine/install/
gcloud CLI	406.0.0	https://cloud.google.com/sdk/docs/install
yq	4.27.5	https://github.com/mikefarah/yq/releases/tag/v4.27.5
kubectrl	1.24	https://kubernetes.io/docs/tasks/tools/install-kubectrl-linux/
oc (only for OpenShift platforms)		https://mirror.openshift.com/pub/openshift-v4/clients/ocp/

Configuration

To configure the TML Cluster on Google Cloud Platform, see [Common Configurations for all Deployment Platforms](#)

Configuring GCP Access in TML Installer

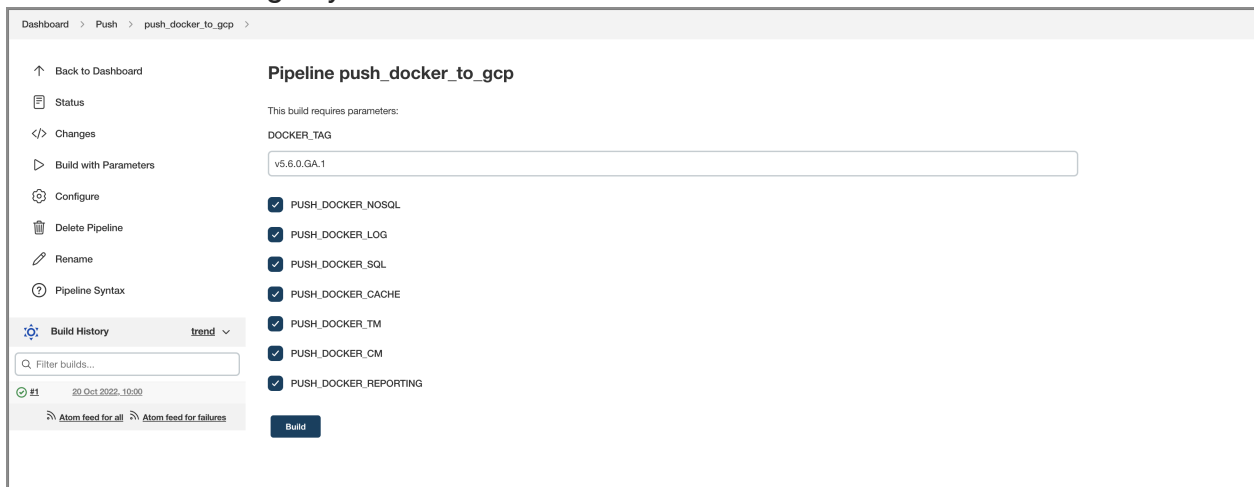
Run Jenkins job "configure_gcp" on **Prepush** tab to configure the access to GCP in TML Installer.



Push TML Docker Images

Before getting started with this step, ensure the images have been built already. If not, run the Jenkins Job "build_docker".

Run Jenkins job "push_docker_to_gcp" on **"Push"** tab to push TML Docker images to Azure container registry.



Parameter	Description
DOCKER_TAG	The tag for TML Docker images, for example: "v5.6.0.GA.1" Note: "5.6.0" is TML release; the last "1" is the build number in Jenkins job "build_docker".
PUSH_DOCKER_NOSQL	Pushes TML Docker image tml-nosql to Azure Container Registry
PUSH_DOCKER_LOG	Pushes TML Docker image tml-log to Azure Container Registry
PUSH_DOCKER_SQL	Pushes TML Docker image tml-sql to Azure Container Registry
PUSH_DOCKER_CACHE	Pushes TML Docker image tml-cache to Azure Container Registry
PUSH_DOCKER_TM	Pushes TML Docker image tml-tm to Azure Container Registry
PUSH_DOCKER_CM	Pushes TML Docker image tml-cm to Azure Container Registry

Deployment on On-premises

Prerequisites

Software Prerequisites

Note: Ensure you have installed the following required softwares and these are available in your PATH before proceeding.

Software	Version Number	Links
Docker	20.x	https://docs.docker.com/engine/install/
yq	4.27.5	https://github.com/mikefarah/yq/releases/tag/v4.27.5
kubectrl	1.24	https://kubernetes.io/docs/tasks/tools/install-kubectrl-linux/
oc (only for OpenShift platforms)		https://mirror.openshift.com/pub/openshift-v4/clients/ocp/

Platform Prerequisites

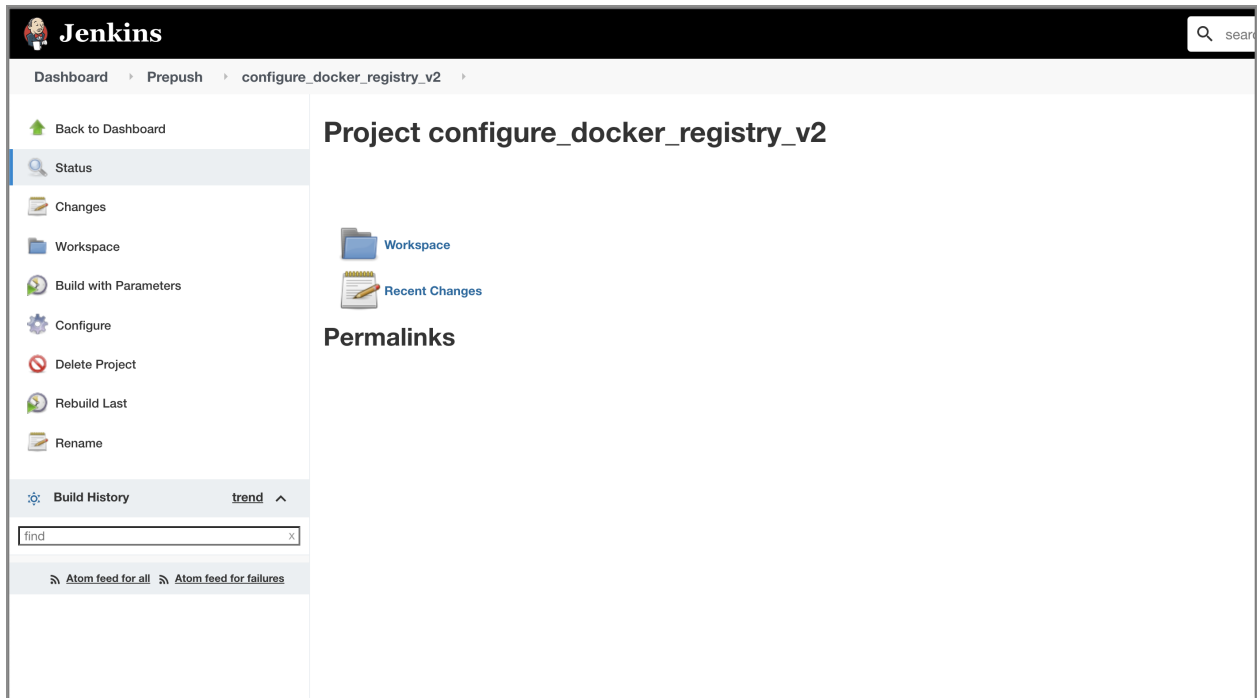
- A running On-prem Kubernetes cluster
- Kubeconfig
- Atleast one Volume Provisioner

Configuration

To configure the TML Cluster on Registry Access, see [Common Configurations for all Deployment Platforms](#)

Configuring Registry Access in TML Installer

Run Jenkins job "configure_docker_registry_v2" on **Prepush** tab to configure the access to Azure in TML Installer.



Name	Definition
DOCKER_REGISTRY_HOST	Internal docker registry to push images
DOCKER_REGISTRY_PORT	The Docker registry port
DOCKER_REGISTRY_USERNAME	Login Username
DOCKER_REGISTRY_PASSWORD	Login Password
DOCKER_REGISTRY_REPO	The repo name inside the registry

Push TML Docker Images

Before getting started with this step, ensure the images have been built already. If not, run the Jenkins Job "build_docker".

Run Jenkins job "push_docker_to_docker_registry_v2" on "Push" tab to push TML Docker images to docker container registry.

Parameter	Description
DOCKER_TAG	The tag for TML Docker images, for example: "v5.6.0.GA.1" Note: "5.6.0" is TML release; the last "1" is the build number in Jenkins job "build_docker".
PUSH_DOCKER_NOSQL	Pushes TML Docker image tml-nosql to Internal Container Registry
PUSH_DOCKER_LOG	Pushes TML Docker image tml-log to Internal Container Registry
PUSH_DOCKER_SQL	Pushes TML Docker image tml-sql to Internal Container Registry
PUSH_DOCKER_CACHE	Pushes TML Docker image tml-cache to Internal Container Registry
PUSH_DOCKER_TM	Pushes TML Docker image tml-tm to Internal Container Registry
PUSH_DOCKER_CM	Pushes TML Docker image tml-cm to Internal Container Registry

Common Configurations for all Deployment Platforms

Planning

Planning and Configuring the TML Cluster

Planning for the TML cluster helps to make following decisions:

- How many logical zones make up your TML cluster?
- How many Traffic Manager, Cache, NoSQL, Log service components would each zone have?

- A minimal TML cluster with 1 Zone and 1 instance of each component requires 4CPU, 16GB memory.
- To support High Availability and Resiliency, you should expect to provide additional resource buffer (CPU, Memory etc.).

Configuration

Run the following command to start the TML Installer:

```
./start-installer.sh DOCKER_HOST_IP
```

Notes:

- DOCKER_HOST_IP is the IP of docker host.
- loopback address (127.0.0.1) **cannot** be used as DOCKER_HOST_IP.

The Jenkins UI is available on **http://<DOCKER_HOST_IP>:8080**

When TML Installer is successfully started, it gives the following message:

```
tml-installer | tml-installer is started.
```

Prebuild

If customer has:

- Boomi-built custom adapter
- custom adapter built from SDK
- portal customization package

then customer admin can use the following Jenkins job to upload above contents before building TML docker images.

Jenkins > Prebuild > [ENABLE AUTO REFRESH](#)

[add description](#)

Prebuild Build Prepush Push Deploy All +

S	W	Name ↓	Last Success	Last Failure	Last Duration
		upload_customer_built_adapter	N/A	N/A	N/A
		upload_mashery_built_adapter	N/A	N/A	N/A
		upload_portal_customization_zip	20 hr - #1	N/A	63 ms

Icon: [S](#) [M](#) [L](#)

[Legend](#) [RSS for all](#) [RSS for failures](#) [RSS for just latest builds](#)

Build Queue

No builds in the queue.

Upload_customer_built_adapter

Jenkins > Prebuild > upload_customer_built_adapter >

[Back to Dashboard](#)

[Status](#)

[Changes](#)

[Workspace](#)

[Build with Parameters](#)

[Delete Project](#)

[Configure](#)

[Rebuild Last](#)

[Rename](#)

Project upload_customer_built_adapter

This build requires parameters:

customer_built_adapter No file selected.
Click "Browse..." button to upload customer-built adapter (.zip)

Customer admin:

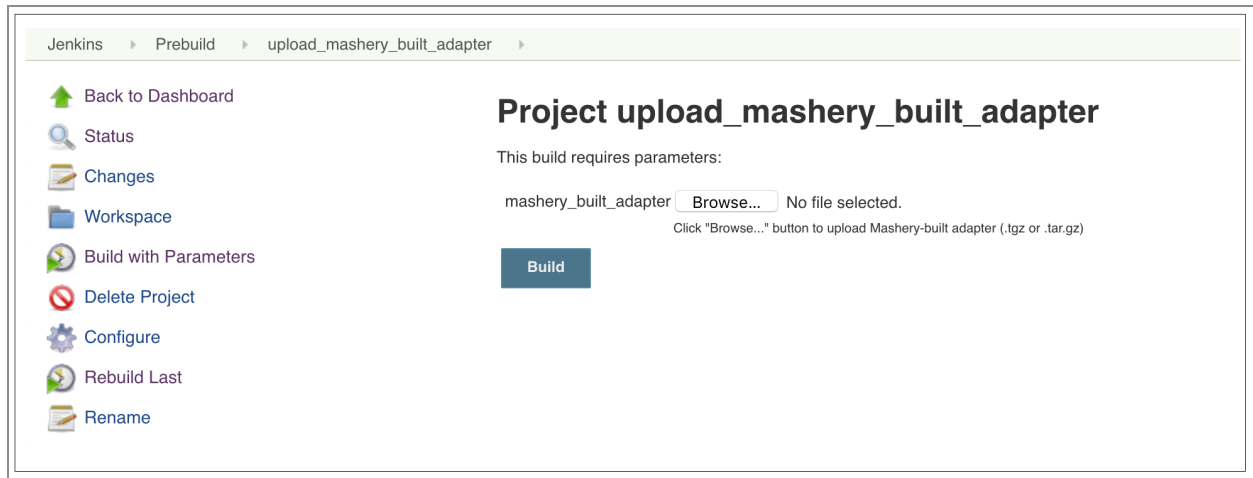
- uploads customer-built adapter in zip.

This Jenkins job:

- saves the customer-built adapter as "/var/jenkins_home/userContent/proxy-extension/customer_extension.zip".

When customer admin builds TML Docker images, the uploaded customer-built adapter will be built into TML Docker image "tml-tm".

upload_mashery_built_adapter



Customer admin:

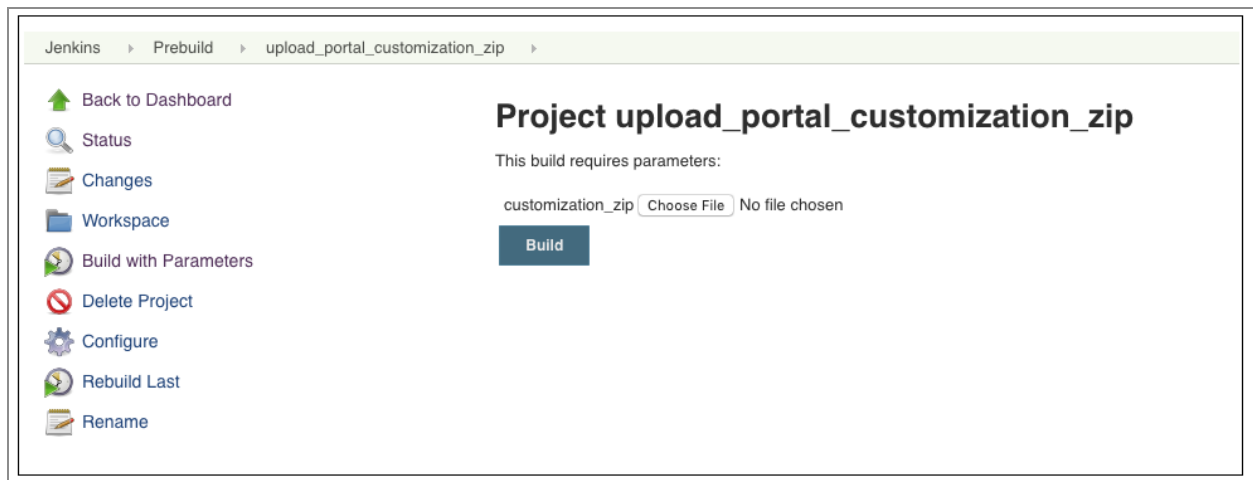
- uploads Boomi-built adapter in tgz.

This Jenkins job:

- extracts the Boomi-built adapter to `"/var/jenkins_home/userContent/proxy-customer"`.

When customer admin builds TML Docker images, the uploaded Boomi-built adapter will be built into TML Docker image `"tml-tm"`.

upload_portal_customization_zip



Customer admin:

- uploads portal customization package in zip.

This Jenkins job:

- saves the customization package as "/var/jenkins_home/userContent/portal/customization.zip".

When customer admin builds TML docker images, the uploaded portal customization package will be built into TML docker image "tml-cm".

Build and Deploy

Creating TML Docker Images

There is one Jenkins job "build_docker" on this tab.

build_docker

Dashboard > Build > build_docker

Project build_docker

This build requires parameters:

RELEASE_VERSION
5.5.0

RELEASE_SUFFIX
GA

☒ BUILD_DOCKER_NOSQL
Build Cassandra docker image.

☒ BUILD_DOCKER_SQL
Build MySQL docker image. (defaults to MySQL CE. For MySQL SE, please refer to documentation).

☒ BUILD_DOCKER_CM
Build Cluster Manager docker image.

☒ BUILD_DOCKER_LOG
Build log service docker image.

☒ BUILD_DOCKER_REPORTING
Build Reporting docker image.

☒ BUILD_DOCKER_CACHE
Build cache docker image.

☒ BUILD_DOCKER_TM
Build Traffic Manager docker image.

TM_CONNECTORS
☒ Connectors available from the installer. Select the connectors to be included into Traffic Manager container.
☐ Connectors uploaded from "upload_mashery_built_adapters".
☐ None.

UPDATE_PACKAGES

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

RELEASE_VERSION

This release version, together with build number, are used to compose Docker image tag.

In the previous image, RELEASE_VERSION is "5.6.0"; on the left pane of screenshot, it shows the build "#1" has been done, so Docker images in build #1 have tag "v5.6.0.GA.1".

Choose Docker Images to Build

Custom admin builds TML Docker images using this Jenkins job:

Parameter	TML Docker Image
BUILD_DOCKER_NOSQL	tml-nosql: NoSQL container
BUILD_DOCKER_LOG	tml-log: Log container
BUILD_DOCKER_SQL	tml-sql: SQL container
BUILD_DOCKER_CACHE	tml-cache: Cache container
BUILD_DOCKER_TM	tml-tm: Traffic Manager container
BUILD_DOCKER_CM	tml-cm: Cluster Manager container

UPDATE_PACKAGES

Customer admin can select:

- **Update_Specified_Packages:** update specified packages during build.
- **Update_All_Packages:** all packages will be updated during build.

update_package_list:

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

Here is an example content of the csv file:

```
python.x86_64
libssh2
systemd
bind-license
openssl-lib
device-mapper.x86_64
device-mapper-lib.x86_64
```

To push images to various registries, see the following options

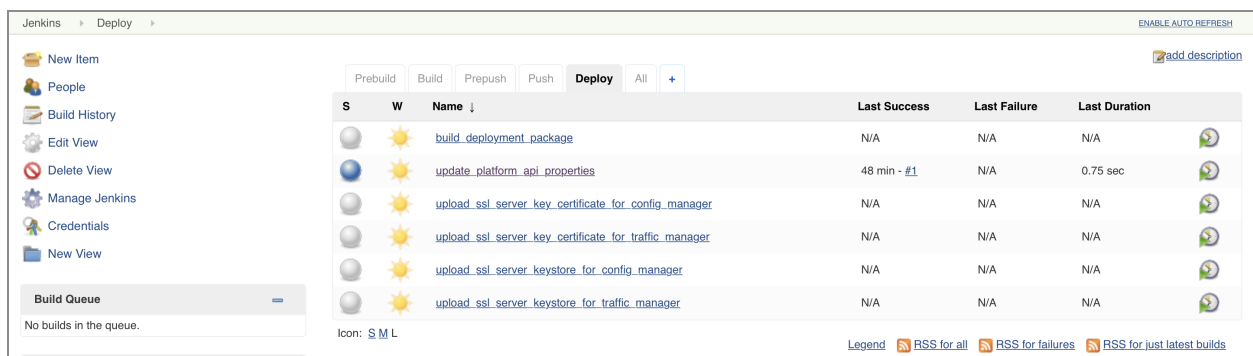
- [Configuring AWS Access in TML Installer](#)
- [Configuring Azure Access in TML Installer](#)
- [Configuring GCP Access in TML Installer](#)
- [Configuring Registry Access in TML Installer](#)

Deployment Customization

Note: If Customer admin skips this customization, the defaults will take effect in Local Edition deployment.

Before deploying the TML cluster, customer admin can

- update Platform API (V3) properties (Jenkins job "update_platform_api_properties")
- customize key stores for Traffic Manager and Configuration Manager (Jenkins jobs with names starting with "upload_ssl_*")

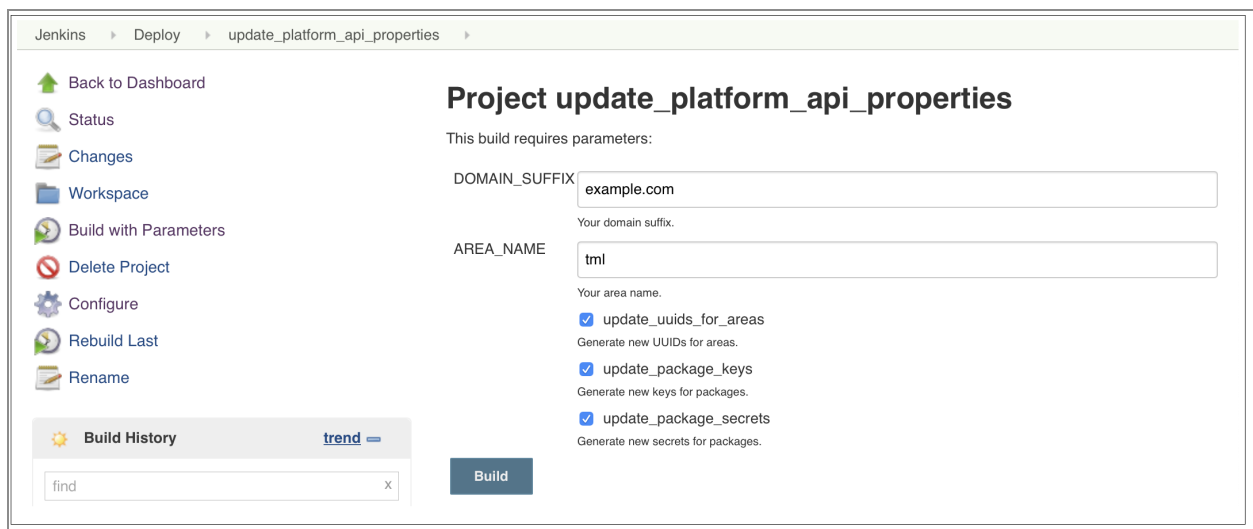


The screenshot shows the Jenkins 'Deploy' view. On the left is a sidebar with navigation links: New Item, People, Build History, Edit View, Delete View, Manage Jenkins, Credentials, and New View. Below this is a 'Build Queue' section indicating 'No builds in the queue.' The main area displays a table of jobs under the 'Deploy' tab. The table has columns for 'S' (Status), 'W' (Weather icon), 'Name', 'Last Success', 'Last Failure', and 'Last Duration'. There are six jobs listed, all with a 'Success' status and a 'Sun' weather icon. The jobs are: 'build_deployment_package', 'update_platform_api_properties', 'upload_ssl_server_key_certificate_for_config_manager', 'upload_ssl_server_key_certificate_for_traffic_manager', 'upload_ssl_server_keystore_for_config_manager', and 'upload_ssl_server_keystore_for_traffic_manager'. The 'update_platform_api_properties' job shows a last success of '48 min - #1' and a duration of '0.75 sec'. At the bottom right, there is a legend for RSS feeds: 'RSS for all', 'RSS for failures', and 'RSS for just latest builds'.

S	W	Name	Last Success	Last Failure	Last Duration
Success	Sun	build_deployment_package	N/A	N/A	N/A
Success	Sun	update_platform_api_properties	48 min - #1	N/A	0.75 sec
Success	Sun	upload_ssl_server_key_certificate_for_config_manager	N/A	N/A	N/A
Success	Sun	upload_ssl_server_key_certificate_for_traffic_manager	N/A	N/A	N/A
Success	Sun	upload_ssl_server_keystore_for_config_manager	N/A	N/A	N/A
Success	Sun	upload_ssl_server_keystore_for_traffic_manager	N/A	N/A	N/A

Update Platform API Properties

For security, customers should update UUIDs for Areas, and update Package Keys and Secrets.



The screenshot shows the Jenkins configuration page for the 'update_platform_api_properties' job. The page title is 'Project update_platform_api_properties'. It indicates that the build requires parameters. There are two input fields: 'DOMAIN_SUFFIX' with the value 'example.com' and 'AREA_NAME' with the value 'tml'. Below these fields, there are three checked checkboxes: 'update_uuids_for_areas' (Generate new UUIDs for areas), 'update_package_keys' (Generate new keys for packages), and 'update_package_secrets' (Generate new secrets for packages). On the left sidebar, there are links for 'Back to Dashboard', 'Status', 'Changes', 'Workspace', 'Build with Parameters', 'Delete Project', 'Configure', 'Rebuild Last', and 'Rename'. At the bottom left, there is a 'Build History' section with a search bar containing 'find' and a 'Build' button.

Using this Jenkins job, customer admin can:

- customize domain suffix
- customize area name
- generate new UUIDs for areas
- generate new keys for packages
- generate new secrets for packages

The file `"/var/jenkins_home/docker-deploy/properties/tml_papi_properties.json"` will be updated.

Preparing KeyStores for Traffic Manager

KeyStores for Traffic Manager can be customized using either the `upload_ssl_server_keystore_for_traffic_manager` or the `upload_ssl_server_key_certificate_for_traffic_manager` Jenkins job.

`upload_ssl_server_keystore_for_traffic_manager`

The screenshot shows the Jenkins web interface for the job `upload_ssl_server_keystore_for_traffic_manager`. The breadcrumb trail is `Jenkins > Deploy > upload_ssl_server_keystore_for_traffic_manager`. On the left sidebar, there are links for `Back to Dashboard`, `Status`, `Changes`, `Workspace`, `Build with Parameters`, `Delete Project`, `Configure`, `Rebuild Last`, and `Rename`. The main content area is titled **Project `upload_ssl_server_keystore_for_traffic_manager`**. Below the title, it says "This build requires parameters:". There are two parameters: `keystore_file` with a `Choose File` button and the text "No file chosen" and "SSL keystore file in PKCS#12 format.", and `keystore_password` with a password input field showing masked characters and the text "SSL keystore password". A `Build` button is located below the parameters.

An Administrator must:

- upload the KeyStore in PKCS#12 format
- input the password protecting the KeyStore

The `upload_ssl_server_keystore_for_traffic_manager` Jenkins job:

- converts the KeyStore to JKS format from PKCS#12 format, saves it as `"/var/jenkins_home/docker-deploy/properties/tml-cm.jks"`;
- exports certificate as `"/var/jenkins_home/docker-deploy/properties/tml-cm-`

crt.pem";

- exports private key as "/var/jenkins_home/docker-deploy/properties/tml-cm-key.pk8";
- records the KeyStore password in "cm_keystore_password" of "/var/jenkins_home/docker-deploy/properties/tml_cm_properties.json".

upload_ssl_server_key_certificate_for_traffic_manager

An Administrator must:

- upload the certificate in PEM format, and
- upload the private key in PKCS#8 format and input the password protecting the private key.

The upload_ssl_server_key_certificate_for_traffic_manager Jenkins job:

- saves the certificate as "/var/jenkins_home/docker-deploy/properties/tml-cm-crt.pem";
- saves the private key as "/var/jenkins_home/docker-deploy/properties/tml-cm-key.pk8";
- puts the certificate and private key to KeyStore in JKS format, saves it as "/var/jenkins_home/docker-deploy/properties/tml-cm.jks";
- records the KeyStore password in "cm_keystore_password" of "/var/jenkins_home/docker-deploy/properties/tml_cm_properties.json".

Preparing KeyStores for Configuration Manager

KeyStores for Configuration Manager can be customized using either the upload_ssl_server_keystore_for_config_manager or the upload_ssl_server_key_certificate_for_

config_manager Jenkins job.

upload_ssl_server_keystore_for_config_manager

Jenkins > Deploy > upload_ssl_server_keystore_for_config_manager

Project upload_ssl_server_keystore_for_config_manager

This build requires parameters:

keystore_file No file chosen
SSL keystore file in PKCS#12 format.

keystore_password
SSL keystore password

An Administrator must:

- upload the KeyStore in PKCS#12 format
- input the password protecting the KeyStore

The upload_ssl_server_keystore_for_config_manager Jenkins job:

- converts the KeyStore to JKS format from PKCS#12 format, then saves it as /var/jenkins_home/docker-deploy/properties/tml-cm.jks
- exports certificate as /var/jenkins_home/docker-deploy/properties/tml-cm-crt.pem
- exports the private key as /var/jenkins_home/docker-deploy/properties/tml-cm-key.pk8
- records the KeyStore password in cm_keystore_password of /var/jenkins_home/docker-deploy/properties/tml_cm_properties.json

upload_ssl_server_key_certificate_for_config_manager

Jenkins > Deploy > upload_ssl_server_key_certificate_for_config_manager

Project upload_ssl_server_key_certificate_for_config_manager

This build requires parameters:

certificate_file No file chosen
SSL certificate file.

key_file No file chosen
Private key file in PKCS#8 format.

key_password
Password protecting key file.

An Administrator must:

- upload the certificate in PEM format
- upload the private key in PKCS#8 format, and input the password protecting the private key

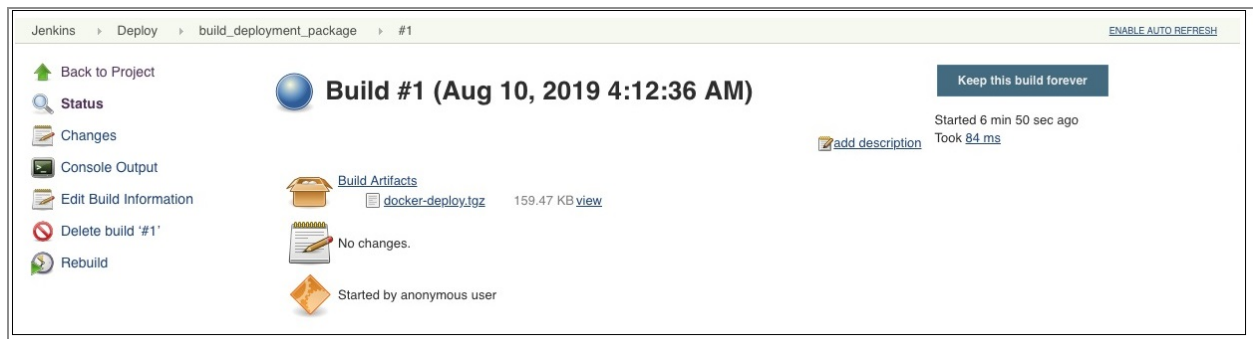
The `upload_ssl_server_key_certificate_for_config_manager` Jenkins job:

- saves the certificate as `/var/jenkins_home/docker-deploy/properties/tml-cm-crt.pem`
- saves the private key as `/var/jenkins_home/docker-deploy/properties/tml-cm-key.pk8`
- converts the certificate and private key to the KeyStore to JKS format, then saves it as `/var/jenkins_home/docker-deploy/properties/tml-cm.jks`
- records the KeyStore password in `cm_keystore_password` in `/var/jenkins_home/docker-deploy/properties/tml_cm_properties.json`

Building Deployment Package

You can either deploy the Local Edition cluster inside the Local Edition installer, or you can build and download the deployment package, then deploy the Local Edition cluster from another host.

To build the deployment package, click **Build Artifacts** in the `build_deployment_package` Jenkins job:



The deployment package, `docker-deploy.tgz`, is available for download from the build archive. The package includes KeyStores, keys, and certificates.

Customizing TML Properties

Note: If Customer admin skips this customization, the defaults will take effect in Local Edition deployment.

From the host where TML Installer is running, run the following commands to login to TML Installer:

```
docker exec -it tml-installer /bin/bash sudo -s
```

or

```
docker exec -it -u root tml-installer /bin/bash
```

Customer admin can customize properties for the TML components in the following way:

Cluster level properties

These are common properties that can apply throughout the entire cluster. For example, MySQL credentials or credentials for MOM sync. These can be customized at `/var/jenkins_home/docker-deploy/properties/tml_cluster_properties.json`.

See Cluster Level Properties

Zone level properties

These are common properties that can apply within a zone, but not necessarily throughout the entire cluster. For example, MOM credentials. These can be customized at `/var/jenkins_home/docker-deploy/properties/tml_zones_properties.json`.

See Zone Level Properties

Component level properties

These are specific properties that apply to specific type of TML components

- Properties for NoSQL component can be customized in `/var/jenkins_home/docker-deploy/properties/tml_nosql_properties.json`.
- Properties for Log Service component can be customized in `/var/jenkins_home/docker-deploy/properties/tml_log_properties.json`. For example, adding log destination details.

See the *Log Management* topic in *Managing a Cluster*.

- Properties for SQL component can be customized in `/var/jenkins_home/docker-deploy/properties/tml_sql_properties.json`.

These properties include parameters for MySQL process as well as the other process running inside the SQL component.

- Properties for Cache component can be customized in `/var/jenkins_home/docker-deploy/properties/tml_cache_properties.json`.

These properties include parameters for the Memcached process as well as other process running inside the Cache component.

- Properties for the Configuration Manager component can be customized in `/var/jenkins_home/docker-deploy/properties/tml_cm_properties.json`.
- Properties for the Traffic Manager component can be customized in `/var/jenkins_home/docker-deploy/properties/tml_tm_properties.json`.

For more information on specific properties, see [this page](#)

After configuring, for deploying on Kubernetes/OpenShift or Docker Swarm platforms, see

- [Deploying on Kubernetes/OpenShift](#)
- [Deploying on Docker Swarm](#)

Deployment on Kubernetes/OpenShift

Prerequisites

- Install the following softwares:

Software	Links
Kubectl CLI	https://kubernetes.io/docs/tasks/tools/
yq	https://github.com/mikefarah/yq/releases/tag/v4.27.5
oc (only for OpenShift platforms)	https://mirror.openshift.com/pub/openshift-v4/clients/ocp/

- Push the Local Edition images created in the container repository.
- Access the deployment manifests inside the tml-installer container in path

```
/var/jenkins_home/docker-deploy
```

or copy to the host machine with the above prerequisites in place

```
docker cp tml-installer:/var/jenkins_home/docker-deploy/ .
```



Note: All the CLI's are preinstalled in tml-installer container

Prerequisites for Openshift

- For Openshift versions after 4.10, enable the NET_RAW and SYS_CHROOT capabilities and increase the pids_limit to 2048.
- cri-o example:
 1. Edit the `crio.conf` file in the `/etc/crio/crio.conf` location.
 2. In the `[crio.runtime]` add `NET_RAW` and `SYS_CHROOT` `default_capabilities = [ "CHOWN", "DAC_OVERRIDE", "FSETID", "FOWNER", "SETGID", "SETUID", "SETPCAP", "NET_BIND_SERVICE", "KILL", "NET_RAW", "SYS_CHROOT", ]`
 3. Restart the cri-o and run the `sudo systemctl restart crio` command.

Access to OpenShift/Kubernetes cluster

Kubernetes

```
export KUBECONFIG=<path to kubeconfig file>
```

Openshift

```
oc login <api server url> -u <username> -p <password>
```

Retrieve Deployment Specific Values

1. Check the namespace where Local Edition needs to be deployed -
 - **Kubernetes**

```
kubectl get namespaces
```

Create a new namespace for APIM LE

```
kubectl create namespace <new namespace name>
```

Switch to the correct/new namespace

```
kubectl config set-context --current --namespace=<namespace name>
```

- **Openshift**

```
oc get project
```

Create a new project

```
oc new-project <new project name>
```

Switch to new project

```
oc project <project name>
```

2. Check for available storage class and choose Kubernetes

```
kubectl get storageclass
```

Openshift

```
oc get storageclass
```

For example, the following is the sample output for Azure:

Sample Output | Azure

NAME	PROVISIONER	RECLAIMPOLICY	VOLUMEBINDINGMODE
managed-csi	disk.csi.azure.com	Delete	WaitForFirstConsumer
3d20h			true

managed-premium (default)	kubernetes.io/azure-disk	Delete	WaitForFirstConsumer
true	3d20h		

Note: Customers can choose different storage classes for different workload types

3. Available Zones in the kubernetes/OpenShift Cluster

Kubernetes

```
kubectl get nodes --show-labels
```

OpenShift

```
oc get nodes --show-labels
```

The following is the sample output:

```
ocp-tml-551-tmal-kpnlp-worker-eastus2-fmkzp Ready worker 3d4h v1.23.5+3afdacb
beta.kubernetes.io/arch=amd64,beta.kubernetes.io/instance-type=Standard_D16s_
v3,beta.kubernetes.io/os=linux,failure-domain.beta.kubernetes.io/region=eastus,failure-
domain.beta.kubernetes.io/zone=eastus-
2,kubernetes.io/arch=amd64,kubernetes.io/hostname=ocp-tml-551-tmal-kpnlp-worker-
eastus2-fmkzp,kubernetes.io/os=linux,node-
role.kubernetes.io/worker=,node.kubernetes.io/instance-type=Standard_D16s_
v3,node.openshift.io/os_id=rhcos,topology.disk.csi.azure.com/zone=eastus-
2,topology.kubernetes.io/region=eastus,topology.kubernetes.io/zone=eastus-2
ocp-tml-551-tmal-kpnlp-worker-eastus3-rmfs2 Ready worker 3d4h v1.23.5+3afdacb
beta.kubernetes.io/arch=amd64,beta.kubernetes.io/instance-type=Standard_D16s_
v3,beta.kubernetes.io/os=linux,failure-domain.beta.kubernetes.io/region=eastus,failure-
domain.beta.kubernetes.io/zone=eastus-
3,kubernetes.io/arch=amd64,kubernetes.io/hostname=ocp-tml-551-tmal-kpnlp-worker-
eastus3-rmfs2,kubernetes.io/os=linux,node-
role.kubernetes.io/worker=,node.kubernetes.io/instance-type=Standard_D16s_
v3,node.openshift.io/os_id=rhcos,topology.disk.csi.azure.com/zone=eastus-
3,topology.kubernetes.io/region=eastus,topology.kubernetes.io/zone=eastus-3
ocp-tml-551-tmal-kpnlp-master-0 Ready master 3d4h v1.23.5+3afdacb
beta.kubernetes.io/arch=amd64,beta.kubernetes.io/instance-type=Standard_D16s_
v3,beta.kubernetes.io/os=linux,failure-domain.beta.kubernetes.io/region=eastus,failure-
domain.beta.kubernetes.io/zone=eastus-
1,kubernetes.io/arch=amd64,kubernetes.io/hostname=ocp-tml-551-tmal-kpnlp-master-
```

```
0,kubernetes.io/os=linux,node-role.kubernetes.io/master=,node.kubernetes.io/instance-
type=Standard_D16s_v3,node.openshift.io/os_
id=rhcos,topology.disk.csi.azure.com/zone=eastus-
1,topology.kubernetes.io/region=eastus,topology.kubernetes.io/zone=eastus-1
```

Create a Image Pull Secret

Kubernetes

```
kubectl create secret docker-registry \
--docker-server=<container registry login url>\
--docker-username=<registry username> \
--docker-password=<registry password> \
tmgc-registry-key
```

Openshift

```
oc create secret docker-registry \
--docker-server=<container registry login url>\
--docker-username=<registry username> \
--docker-password=<registry password> \
tmgc-registry-key
```

Check if key gets created successfully

```
oc get secrets
```

NAME	TYPE	DATA	AGE
tmgc-registry-key	kubernetes.io/dockerconfigjson	1	3d20h

Customize the TML Deployment Manifest

You can create a kubernetes cluster spanning one or more nodes based on the outcome of the planning phase. Also, change the deployment manifest appropriately with the number of components per zone.

In folder, `"/var/jenkins_home/docker-deploy/kubernetes/manifest.yaml"` of TML Installer, customize the TML Deployment Manifest

Note: Before you customize your API Management - Local Edition cluster, ensure the `tml_image_tag` matches your Docker image build. For example, to verify `"tml_image_tag": "v5.6.0.GA.1"`, you can check the Docker image build in Jenkins as shown below:

The top screenshot shows the Jenkins configuration page for 'Project tmgc-build-docker'. The 'RELEASE_VERSION' parameter is set to '5.0.0'. The bottom screenshot shows the 'Build #1' details page, which includes a list of build artifacts. A red box highlights 'Build #1 (Sep 14, 2018 12:02:46 AM)' and a red arrow points from the '5.0.0' in the top screenshot to the 'v5.0.0.1' tag in the build artifacts list.

Following is a Manifest example:

```
---
# APIM-LE Cluster Name
tml_cluster_name: reference-tml-cluster-name

# APIM-LE Cluster Mode; Values - untethered, tethered
tml_cluster_mode: untethered

# APIM-LE Deployment Zones
# Example values -
# AWS: us-east-1a,us-east-1c
```

```

# Azure: eastus-1, eastus-2
# GCP: us-central1-a, us-central1-b
tml_zones:
- zone-1
- zone-2

# Cluster Platform
# Values: k8s/ops
tml_cluster_platform: k8s

# Image & Registry details
tml_image_tag: v5.6.0.GA.1
tml_image_registry_host: tml-registry-host
tml_image_repo: tml-image-repo

# Component Replica Counts
tml_cm_count: 1
tml_tm_count: 3
tml_cache_count: 3
tml_sql_count: 3
tml_log_count: 2
tml_nosql_count: 3

# Security Context Constraints
# For Openshift use tml_username, for k8s platforms use tml_userid
tml_username: default
tml_userid: '1001'

# Namespace for Cluster Component Deployment
tml_deploy_namespace: default

# Storage Properties for APIM LE
# set storage size and storage name for each component type
# Default Storage Class Names per platform
# AWS - gp2
# GCP - standard-rwo
# Azure - managed-premium
tml_sql_storage_size: 10Gi
tml_sql_storage_name: apim-le-sql-storageclass
tml_log_storage_size: 10Gi
tml_log_storage_name: apim-le-log-storageclass
tml_nosql_storage_size: 10Gi
tml_nosql_storage_name: apim-le-nosql-storageclass
tml_cache_storage_size: 10Gi
tml_cache_storage_name: apim-le-cache-storageclass
tml_reporting_storage_size: 10Gi
tml_reporting_storage_name: apim-le-reporting-storageclass

```

```
# Configure component ports and protocol
tml_tm_http_enabled: 'true'
tml_tm_http_port: 80
tml_tm_https_enabled: 'true'
tml_tm_https_port: 443
tml_tm_mhttps_enabled: 'false'
tml_tm_mhttps_port: 1443
tml_tm_oauth_enabled: 'true'
tml_tm_oauth_port: 8083
tml_tm_active_retries: 90
tml_api_http_enabled: 'true'
tml_api_http_port: 7080
tml_api_https_enabled: 'true'
tml_api_https_port: 7443
tml_cm_http_enabled: 'true'
tml_cm_http_port: 8080
tml_cm_https_enabled: 'true'
tml_cm_https_port: 8443
tml_sql_restore: 'false'
cassandra_max_heap: 512M
cassandra_replication_factor: 3

# Enable Reporting? Default - false
tml_reporting_enabled: 'false'
tml_reporting_http_enabled: 'true'
tml_reporting_http_port: 3000
```

The following parameters must be customized as per your requirements:

Parameter Name	Parameter Value	Note
tml_cluster_name	tmlK8SCluster	MUST customize: • camel is the naming convention; • no space or special character.
tml_cluster_mode	tethered untethered	In "tethered" mode, admin must specify "mom_key" and "mom_secret" in "/var/jenkins_home/docker-deploy/properties/tml_cluster_properties.json"

Parameter Name	Parameter Value	Note
tml_zones	Example: - us-east-1a - us-east-1c	
tml_image_registry_host	container registry login url	customizable: MUST be consistent with "configure_<platform>" on "Prepush".
tml_image_repo	Base folder path where all images are pushed.	Default - tml
tml_image_tag	TML docker image tag	MUST customize to match the docker build to be deployed: i.e. the "5.6.0" is TML release; the last "1" is the build number in Jenkins job "build_docker".
tml_sql_storage_name		MUST customize to match one of the configured storage classes in the K8s/Openshift Cluster
tml_log_storage_name		
tml_nosql_storage_name		
tml_cache_storage_name		
tml_reporting_storage_name		

**Note:**

In tethered mode, the `tml_sql_count` should be set to 1.



Note: For On Prem Kubernetes/OpenShift platforms, the `tml_sql_count` should be set to 1.

Generate Deployment Scripts

For example, generate the deployment scripts, in folder `"/var/jenkins_home/docker-deploy/kubernetes/"` of TML Installer and run the following command to generate deployment scripts:

```
./compose.sh manifest.yaml
```

The following is an output example:

```
TMG_CLUSTER_NAME: reference-tml-cluster-name
TMG_CLUSTER_MODE: tethered
in "tethered" mode, admin must specify "api_key" and "api_secret" in "tml_sql_properties.json" and
"tml_log_properties.json"
TMG_CLUSTER_TAG: 4984199f97988b245f35601743b9303b
K8S_AWS_ZONES us-east-1e
TMG_ZONE_NAME 0: us-east-1e
TMG_ZONE_COUNT: 1
TMG_RELEASE_VERSION: 5.6.0
TMGC_REPO_NAME: tml/v5.6.0.GA.1
repositoryUri 894547326815.dkr.ecr.us-east-1.amazonaws.com/tml/v5.6.0.GA.1
AWS_REGISTRY_HOST 894547326815.dkr.ecr.us-east-1.amazonaws.com
```

Create TML Cluster

For example, create the TML cluster in a single zone, in folder `"/var/jenkins_home/docker-deploy/kubernetes/manifest/"` and run the following command to create TML cluster:

```
./create-tml-cluster.sh
```

Validate TML Cluster

If the deployment is still running, you can check on the progress using the following steps:

1. Check the running pods

```
$ kubectl get pods
```

2. Log in to the CM container and check the components and their states

```
$ kubectl exec -ti cm-deploy-0-7b6b96bbf5-w5w22 bash
```

3. You are now logged in to the CM container. Check the components and their states by running the following commands.

```
[restricted@cm-deploy-0-7b6b96bbf5-w5w22 builder]# alias cm=clustermanager
[restricted@cm-deploy-0-7b6b96bbf5-w5w22 builder]# cm ls components
Using cluster [Tibco Mashery Local Reference Cluster]
Using Zone [local]
Component ID          Component Type  Component Name  Component Status  Component
Host    Component Agent Port  Component Service Port(s)
-----
b38efc8f-bffa-4a68-bcf5-a21370434b78 cache          cache
ACTIVE          10.0.0.12908011212,11211,11213,11214,11215,11216
f1d75242-4c67-4eb9-a8b3-63195689410c configmanager  cm
RESOLVED          10.0.0.690807080
11261544-ef93-4d1f-8da9-af71b1593763 logservice      log          ACTIVE          10.0.0.8908024224
66157d26-43b5-40ff-aefe-b1f52aa2f44b nosql           nosql        ACTIVE          10.0.0.390809042
fa66dcbb-35af-4fb3-bc87-09382c0f20ef sql            sql          ACTIVE          10.0.0.1090803306
84792b97-b4ba-4a28-b7f4-1b5cfbd6c76a trafficmanager  tm
STANDBY          10.0.0.1490808080
```

4. Check the status of the overall cluster and see what action is pending

```
[restricted@cm-deploy-0-7b6b96bbf5-w5w22 builder]# cm cluster status
*** Checking the status of Cluster 'Tibco Mashery Local Reference Cluster'***
*** Checking the status of zone 'local'***
TML component [66157d26-43b5-40ff-aefe-b1f52aa2f44b] of type [nosql] found with status [ACTIVE] on
Host IP [10.0.0.3]..
TML component [11261544-ef93-4d1f-8da9-af71b1593763] of type [logservice] found with status [ACTIVE]
on Host IP [10.0.0.8]..
TML component [fa66dcbb-35af-4fb3-bc87-09382c0f20ef] of type [sql] found with status [ACTIVE] on Host
IP [10.0.0.10]..
TML component [b38efc8f-bffa-4a68-bcf5-a21370434b78] of type [cache] found with status [ACTIVE] on
```

```
Host IP [10.0.0.12]..
TML component [84792b97-b4ba-4a28-b7f4-1b5cfbd6c76a] of type [trafficmanager] found with status
[STANDBY] on Host IP [10.0.0.14]. See details below
Component ID      Component Name  Component IP    Component Status  Pending Activity
Error
-----
84792b97-b4ba-4a28-b7f4-1b5cfbd6c76a tm            10.0.0.14Not Ready
```

5. In some time, the cluster status should turn READY. You can check the last failing component for its individual status as follows. In this case, the Traffic Manager running on 10.0.0.14 IP

```
[root@5a527d7ef2abbuilder]# curl http://10.0.0.14:9080/container/status
{
  "tmgcld": "84792b97-b4ba-4a28-b7f4-1b5cfbd6c76a",
  "ip": "10.0.0.14",
  "name": "tm",
  "pending_activity": "checking component status ...",
  "status": "Ready"
}
```

6. As seen above, by now the Traffic Manager status is ready. The pending activity was the last successful activity performed by the container agent.
7. If the status indicates error, then please check the log-service container for corresponding errors.
 - a. Log in to the logservice container
 - b. Check the appropriate log files under the **"/mnt/data"** location
 - c. For example, to check any errors in the CM container, check the **"/mnt/data/tml-cm"** location.

Deployment on Docker Swarm

Prerequisites

i Note: Ensure you have installed the below required softwares and the are available in your PATH before proceeding

- Install the following software.

Software	Links
yq	https://github.com/mikefarah/yq/releases/tag/v4.27.5

- Push the APIM LE images created to container repository.
- Access the deployment manifests the tml-installer container in path

```
/var/jenkins_home/docker-deploy/docker-swarm
```

Files to modify

Navigate to the folder **/docker-deploy/docker-swarm/** in the installer. You need to update the reference manifest deployment file **manifest.yaml** for customizing your deployment.

TML Cluster Configuration

Below are the cluster level properties:

"tml_cluster_name": "TIBCO Mashery Gateway Reference Cluster",

"tml_cluster_mode": "untethered",

"tml_zones": ["local"],



Note: Ensure you specify a valid value for "tml_cluster_mode". Only 2 values are valid:

1. **tethered** : In this mode, ML gets the configuration data from ML cloud (You need to have MOM Key and Secret in this mode).
2. **untethered** : In this mode, you configure data through ML config manager UI.

TML Component configuration

The following settings can be customized.

tml_image_tag : Docker image tag in the form of v5.2.0.<build no> . Example : v5.2.0.10

tml_image_registry_host : The docker registry where tml images were pushed.

tml_image_repo : TML image repo. Default is "tml".

i Note:

Variable prefix	Description
tml_cm_count	Cluster Manager Component count
tml_tm_count	Traffic Manager Component count
tml_cache_count	Cache Component count
tml_sql_count	SQL Component count
tml_log_count	Log Component count
tml_nosql_count	NoSQL (Cassandra) Component count

Sample Manifest

```
# APIM-LE Cluster Name
tml_cluster_name: Tibco Mashery Local Reference Cluster

# APIM-LE Cluster Mode; Values - untethered, tethered
tml_cluster_mode: untethered

# APIM-LE Deployment Zones
tml_zones:
  - local

# Image & Registry details
tml_image_registry_host: tml-registry-host
tml_image_tag: v5.6.0.GA.1
tml_image_repo: tml-image-repo-name

# Component Replica Count per zone
tml_cm_count: 1
tml_tm_count: 1
tml_cache_count: 1
tml_sql_count: 1
tml_log_count: 1
tml_nosql_count: 1
```

```
# Service UID for Component Containers
tml_userid: "1001"tml_tm_http_enabled: "true"tml_tm_http_port: 80
tml_tm_https_enabled: "true"tml_tm_https_port: 443
tml_tm_mhttps_enabled: "false"tml_tm_mhttps_port: 1443
tml_tm_oauth_enabled: "true"tml_tm_oauth_port: 8083
tml_tm_active_retries: 90
tml_api_http_enabled: "true"tml_api_http_port: 7080
tml_api_https_enabled: "true"tml_api_https_port: 7443
tml_cm_http_enabled: "true"tml_cm_http_port: 8080
tml_cm_https_enabled: "true"tml_cm_https_port: 8443
cassandra_max_heap: 512M
cassandra_replication_factor: 1

# Enable Reporting? Default - false
tml_reporting_enabled: "false"tml_reporting_http_enabled: "true"tml_reporting_http_port: 3000
```

Setting up Mashery Service Configuration data

Option 1: Using TML Config Manager

You must set "tml_cluster_mode" as **"untethered"** in docker-deploy/docker-swarm/manifest.yaml file

Option 2: Synchronizing Service Configuration data from Mashery On-premise Manager (MOM)

TML also offers the capability to sync data from an active mom host at the time of cluster creation. To load the data using the MoM sync configuration please specify the following three properties in the docker-deploy/properties/tml_cluster_properties.json.

```
"mom_key" : "<MOM_API_KEY>",
"mom_secret" : "<MOM_API_SECRET>",
```

The tml_sql_properties.json is located in the docker-deploy/properties/ directory.



Note: You must set " tml_cluster_mode " as " tethered " in docker-deploy/docker-swarm/manifest.yaml file.

Generate the deployment scripts

Run the following command and find generated deployment scripts and configuration in the folder "manifest":

```
./compose.sh manifest.yaml
```

Create TML Cluster

Navigate to the folder **docker-deploy/docker-swarm/manifest**.

1. Login to the docker registry.

```
docker login <registry host>
-u <user> -p <password>
```



Note: Ensure log in is successful before moving ahead.

2. Get the node name from the docker host

Node details

```
docker node ls
```

Sample Output

ID	HOSTNAME	STATUS	AVAILABILITY	MANAGER STATUS
ENGINE VERSION				
y23nlj594bdaeuz5716u42mdz *	na2devmasrc101	Ready	Active	Leader 20.10.9

3. Export HOST_NAME to provide node details

Sample

```
export HOST_NAME=na2devmasrc101
```

4. Run "create-tml-cluster.sh"

Create TML Cluster

```
./create-tml-cluster.sh
```


Sample output

```

Creating overlay network ml5 for the deployment.
rdrzodc6pt24q82n961o0wuww
Deploying cassandra pods.
Creating secret nosqlstack_nosql-p12-vol
Creating secret nosqlstack_nosql-trust-p12-vol
Creating secret nosqlstack_nosql_resource
Creating secret nosqlstack_nosql_config
Creating secret nosqlstack_cluster-property-vol
Creating secret nosqlstack_zones-property-vol
Creating service nosqlstack_nosql_seed
vnt1pw2p8317 nosqlstack_nosql_seed replicated 0/1
na2devmashbr01.dev.tibco.com/tml/tml-nosql:v5.6.0.GA.1
vnt1pw2p8317 nosqlstack_nosql_seed replicated 0/1
na2devmashbr01.dev.tibco.com/tml/tml-nosql:v5.6.0.GA.1
vnt1pw2p8317 nosqlstack_nosql_seed replicated 0/1
na2devmashbr01.dev.tibco.com/tml/tml-nosql:v5.6.0.GA.1
vnt1pw2p8317 nosqlstack_nosql_seed replicated 0/1
na2devmashbr01.dev.tibco.com/tml/tml-nosql:v5.6.0.GA.1
vnt1pw2p8317 nosqlstack_nosql_seed replicated 0/1
na2devmashbr01.dev.tibco.com/tml/tml-nosql:v5.6.0.GA.1
vnt1pw2p8317 nosqlstack_nosql_seed replicated 0/1
na2devmashbr01.dev.tibco.com/tml/tml-nosql:v5.6.0.GA.1
vnt1pw2p8317 nosqlstack_nosql_seed replicated 0/1
na2devmashbr01.dev.tibco.com/tml/tml-nosql:v5.6.0.GA.1
vnt1pw2p8317 nosqlstack_nosql_seed replicated 1/1
na2devmashbr01.dev.tibco.com/tml/tml-nosql:v5.6.0.GA.1
Deploying CM pods.
Creating secret cmstack_cm-jks-vol
Creating secret cmstack_cm-crt-vol
Creating secret cmstack_cm-key-vol
Creating secret cmstack_cm_config
Creating secret cmstack_cm_resource
Creating secret cmstack_cluster-property-vol
Creating secret cmstack_zones-property-vol
Creating secret cmstack_papi-property-vol
Creating service cmstack_tm1cm
t4gvzy4upke6 cmstack_tm1cm replicated 0/1
na2devmashbr01.dev.tibco.com/tml/tml-cm:v5.6.0.GA.1 *:7080->7080/tcp, *:7443-
>7443/tcp, *:8080->8080/tcp, *:8443->8443/tcp
t4gvzy4upke6 cmstack_tm1cm replicated 0/1

```

```

na2devmashbr01.dev.tibco.com/tml/tml-cm:v5.6.0.GA.1 *:7080->7080/tcp, *:7443-
>7443/tcp, *:8080->8080/tcp, *:8443->8443/tcp
t4gvzy4upke6 cmstack_tmlcm replicated 1/1
na2devmashbr01.dev.tibco.com/tml/tml-cm:v5.6.0.GA.1 *:7080->7080/tcp, *:7443-
>7443/tcp, *:8080->8080/tcp, *:8443->8443/tcp
Creating secret logstack_zones-property-vol
Creating secret logstack_log_config
Creating secret logstack_log_resource
Creating secret logstack_cluster-property-vol
Creating service logstack_log

va9zqskyfiit logstack_log replicated 0/1
na2devmashbr01.dev.tibco.com/tml/tml-log:v5.6.0.GA.1

va9zqskyfiit logstack_log replicated 0/1
na2devmashbr01.dev.tibco.com/tml/tml-log:v5.6.0.GA.1

va9zqskyfiit logstack_log replicated 0/1
na2devmashbr01.dev.tibco.com/tml/tml-log:v5.6.0.GA.1

va9zqskyfiit logstack_log replicated 0/1
na2devmashbr01.dev.tibco.com/tml/tml-log:v5.6.0.GA.1

va9zqskyfiit logstack_log replicated 1/1
na2devmashbr01.dev.tibco.com/tml/tml-log:v5.6.0.GA.1
Deploying DB pods.
Creating secret sqlstack_cluster-property-vol
Creating secret sqlstack_zones-property-vol
Creating secret sqlstack_papi-property-vol
Creating secret sqlstack_sql_resource
Creating secret sqlstack_sql_config
Creating service sqlstack_sql
6drvcapzkxpd sqlstack_sql replicated 0/1
na2devmashbr01.dev.tibco.com/tml/tml-sql:v5.6.0.GA.1
6drvcapzkxpd sqlstack_sql replicated 0/1
na2devmashbr01.dev.tibco.com/tml/tml-sql:v5.6.0.GA.1
6drvcapzkxpd sqlstack_sql replicated 0/1
na2devmashbr01.dev.tibco.com/tml/tml-sql:v5.6.0.GA.1
6drvcapzkxpd sqlstack_sql replicated 0/1
na2devmashbr01.dev.tibco.com/tml/tml-sql:v5.6.0.GA.1
6drvcapzkxpd sqlstack_sql replicated 0/1
na2devmashbr01.dev.tibco.com/tml/tml-sql:v5.6.0.GA.1
6drvcapzkxpd sqlstack_sql replicated 0/1
na2devmashbr01.dev.tibco.com/tml/tml-sql:v5.6.0.GA.1
6drvcapzkxpd sqlstack_sql replicated 0/1
na2devmashbr01.dev.tibco.com/tml/tml-sql:v5.6.0.GA.1

```

```

6drvcapzkxpd sqlstack_sql replicated 0/1
na2devmashbr01.dev.tibco.com/tml/tml-sql:v5.6.0.GA.1
6drvcapzkxpd sqlstack_sql replicated 1/1
na2devmashbr01.dev.tibco.com/tml/tml-sql:v5.6.0.GA.1
Deploying memcache pods.
Creating secret cachestack_cache_config
Creating secret cachestack_cache_resource
Creating secret cachestack_cluster-property-vol
Creating secret cachestack_zones-property-vol
Creating service cachestack_cache

k8k85p6kgvks cachestack_cache replicated 0/1
na2devmashbr01.dev.tibco.com/tml/tml-cache:v5.6.0.GA.1

k8k85p6kgvks cachestack_cache replicated 0/1
na2devmashbr01.dev.tibco.com/tml/tml-cache:v5.6.0.GA.1

k8k85p6kgvks cachestack_cache replicated 1/1
na2devmashbr01.dev.tibco.com/tml/tml-cache:v5.6.0.GA.1
Deploying TM pods.
Creating secret tmstack_tm-jks-vol
Creating secret tmstack_tm-trust-jks-vol
Creating secret tmstack_tm_resource
Creating secret tmstack_tm_config
Creating secret tmstack_cluster-property-vol
Creating secret tmstack_zones-property-vol
Creating service tmstack_tm

xrr5gv9bw9s7 tmstack_tm replicated 0/1
na2devmashbr01.dev.tibco.com/tml/tml-tm:v5.6.0.GA.1 *:80->20080/tcp, *:443-
>20443/tcp, *:8083->8083/tcp

xrr5gv9bw9s7 tmstack_tm replicated 0/1
na2devmashbr01.dev.tibco.com/tml/tml-tm:v5.6.0.GA.1 *:80->20080/tcp, *:443-
>20443/tcp, *:8083->8083/tcp

xrr5gv9bw9s7 tmstack_tm replicated 0/1
na2devmashbr01.dev.tibco.com/tml/tml-tm:v5.6.0.GA.1 *:80->20080/tcp, *:443-
>20443/tcp, *:8083->8083/tcp

xrr5gv9bw9s7 tmstack_tm replicated 0/1
na2devmashbr01.dev.tibco.com/tml/tml-tm:v5.6.0.GA.1 *:80->20080/tcp, *:443-
>20443/tcp, *:8083->8083/tcp

```

```
xrr5gv9bw9s7 tmstack_tm replicated 0/1
na2devmashbr01.dev.tibco.com/tml/tml-tm:v5.6.0.GA.1 *:80->20080/tcp, *:443-
>20443/tcp, *:8083->8083/tcp
```

```
xrr5gv9bw9s7 tmstack_tm replicated 0/1
na2devmashbr01.dev.tibco.com/tml/tml-tm:v5.6.0.GA.1 *:80->20080/tcp, *:443-
>20443/tcp, *:8083->8083/tcp
```

```
xrr5gv9bw9s7 tmstack_tm replicated 0/1
na2devmashbr01.dev.tibco.com/tml/tml-tm:v5.6.0.GA.1 *:80->20080/tcp, *:443-
>20443/tcp, *:8083->8083/tcp
```

```
xrr5gv9bw9s7 tmstack_tm replicated 0/1
na2devmashbr01.dev.tibco.com/tml/tml-tm:v5.6.0.GA.1 *:80->20080/tcp, *:443-
>20443/tcp, *:8083->8083/tcp
```

```
xrr5gv9bw9s7 tmstack_tm replicated 0/1
na2devmashbr01.dev.tibco.com/tml/tml-tm:v5.6.0.GA.1 *:80->20080/tcp, *:443-
>20443/tcp, *:8083->8083/tcp
```

```
xrr5gv9bw9s7 tmstack_tm replicated 0/1
na2devmashbr01.dev.tibco.com/tml/tml-tm:v5.6.0.GA.1 *:80->20080/tcp, *:443-
>20443/tcp, *:8083->8083/tcp
```

```
xrr5gv9bw9s7 tmstack_tm replicated 0/1
na2devmashbr01.dev.tibco.com/tml/tml-tm:v5.6.0.GA.1 *:80->20080/tcp, *:443-
>20443/tcp, *:8083->8083/tcp
```

```
xrr5gv9bw9s7 tmstack_tm replicated 0/1
na2devmashbr01.dev.tibco.com/tml/tml-tm:v5.6.0.GA.1 *:80->20080/tcp, *:443-
>20443/tcp, *:8083->8083/tcp
```

```
xrr5gv9bw9s7 tmstack_tm replicated 0/1
na2devmashbr01.dev.tibco.com/tml/tml-tm:v5.6.0.GA.1 *:80->20080/tcp, *:443-
>20443/tcp, *:8083->8083/tcp
```

```
xrr5gv9bw9s7 tmstack_tm replicated 0/1
na2devmashbr01.dev.tibco.com/tml/tml-tm:v5.6.0.GA.1 *:80->20080/tcp, *:443-
>20443/tcp, *:8083->8083/tcp
```

```
xrr5gv9bw9s7 tmstack_tm replicated 0/1
na2devmashbr01.dev.tibco.com/tml/tml-tm:v5.6.0.GA.1 *:80->20080/tcp, *:443-
>20443/tcp, *:8083->8083/tcp
```

```
xrr5gv9bw9s7 tmstack_tm replicated 1/1
na2devmashbr01.dev.tibco.com/tml/tml-tm:v5.6.0.GA.1 *:80->20080/tcp, *:443->20443/tcp, *:8083->8083/tcp
```

Validate TML Cluster

Verify Swarm and TML cluster

1. Run below command to check if correct no of containers have been created. If no of nosql container is greater than 1, then one will be created as seed node and remaining as non seed.

docker service ls

Sample output:

ID	NAME	MODE	REPLICAS	IMAGE	PORTS
k8k85p6kgvks	cachestack_cache	replicated	1/1	na2devmashbr01.dev.tibco.com/tml/tml-cache:v5.6.0.GA.1	
t4gvzy4upke6	cmstack_tmldm	replicated	1/1	na2devmashbr01.dev.tibco.com/tml/tml-cm:v5.6.0.GA.1	*:7080->7080/tcp, *:7443->7443/tcp, *:8080->8080/tcp, *:8443->8443/tcp
va9zqskyfiit	logstack_log	replicated	1/1	na2devmashbr01.dev.tibco.com/tml/tml-log:v5.6.0.GA.1	
vnt1pw2p8317	nosqlstack_nosql_seed	replicated	1/1	na2devmashbr01.dev.tibco.com/tml/tml-nosql:v5.6.0.GA.1	
6drvapzkxpd	sqlstack_sql	replicated	1/1	na2devmashbr01.dev.tibco.com/tml/tml-sql:v5.6.0.GA.1	
xrr5gv9bw9s7	tmstack_tm	replicated	1/1	na2devmashbr01.dev.tibco.com/tml/tml-tm:v5.6.0.GA.1	*:80->20080/tcp, *:443->20443/tcp, *:8083->8083/tcp

2. You can also get the placement of containers node wise by running below command.

docker node ps \$(docker node ls -q) --filter desired-state=Running | uniq

Sample output:

ID	NAME	IMAGE	NODE	DESIRED

STATE	CURRENT STATE	ERROR	PORTS
2cxvfhyzcey0	cachestack_cache.1	na2devmashbr01.dev.tibco.com/tml/tml-cache:v5.6.0.GA.1	na2devmasasc01 Running Running 9 minutes ago
6sjbfzdsbmjk	cmstack_tmlcm.1	na2devmashbr01.dev.tibco.com/tml/tml-cm:v5.6.0.GA.1	na2devmasasc01 Running Running 10 minutes ago
ltx8ra7n0emq	logstack_log.1	na2devmashbr01.dev.tibco.com/tml/tml-log:v5.6.0.GA.1	na2devmasasc01 Running Running 10 minutes ago
svhssvg7dadb	nosqlstack_nosql_seed.1	na2devmashbr01.dev.tibco.com/tml/tml-nosql:v5.6.0.GA.1	na2devmasasc01 Running Running 11 minutes ago
o25o8jb04m9d	sqlstack_sql.1	na2devmashbr01.dev.tibco.com/tml/tml-sql:v5.6.0.GA.1	na2devmasasc01 Running Running 10 minutes ago
qbexj0owawjl	tmstack_tm.1	na2devmashbr01.dev.tibco.com/tml/tml-tm:v5.6.0.GA.1	na2devmasasc01 Running Running 3 minutes ago

3. To examine the current state of cluster, go the cm container. Make note of node where cm is deployed. You can open another terminal and configure docker client to talk to that node. If tml-cm container is deployed on node-1, then open a new terminal and run below commands in sequence.
 - a. `eval "$(docker-machine env node-1)"`
 - b. `docker ps`
 - c. `docker exec -it <cm container id> /bin/bash`
 - d. You will see the list of commands you can execute within the container. Here is a common set of commands you may want to use

```

1. [root@51ffa1197313 builder]# alias cm=clustermanager

2. [root@51ffa1197313 builder]# cm ls clusters
Cluster ID          Cluster Name
-----
b353e255-4ea6-4403-af2d-2c5e72ccc2f4 Tibco Mashery Loc...

3. [root@51ffa1197313 builder]# cm use cluster b353e255-4ea6-4403-af2d-2c5e72ccc2f4
Using cluster [Tibco Mashery Local Reference Cluster]

4. [root@51ffa1197313 builder]# cm ls zones
Using cluster [Tibco Mashery Local Reference Cluster]
Zone ID          Zone Name
-----
c70f8e9f-d65d-44b7-8569-2523fd66998d local

5. [root@51ffa1197313 builder]# cm use zone c70f8e9f-d65d-44b7-8569-2523fd66998d
Using cluster [Tibco Mashery Local Reference Cluster]
```

Using Zone [local]

6. [root@51ffa1197313 builder]# cm ls components

Using cluster [Tibco Mashery Local Reference Cluster]

Using Zone [local]

Component ID	Component Type	Component Name	Component Status	
Component Host	Component Agent Port	Component Service Port(s)		
65686417-60a7-46b0-8195-8a57a7ff61e1	cache	cache	ACTIVE	10.0.0.13
9080	11212,11211,11213,11214,11215,11216			
1dee1137-6e96-4aad-a110-7a3ee6934717	configmanager	cm	ACTIVE	
10.0.0.7	9080	8080		
92445f5c-a540-45c7-adbb-9a50475a36bb	logservice	log	ACTIVE	10.0.0.9
9080	24224			
f598a428-d846-46e8-9e46-10944cbc31e6	nosql	nosql	ACTIVE	10.0.0.5
9080	9042			
be83b246-5ea6-4559-80c8-b185313cad47	sql	sql	ACTIVE	10.0.0.11
9080	3306			
c5da4ef1-2531-430b-b832-a9617d3bd6a5	trafficmanager	tm	ACTIVE	10.0.0.15
9080				

Making a traffic call

1. In docker swarm cluster, you can use any machine's IP to make traffic call. In the current working terminal, you can the master IP using below command.

```
MASTER_HOST_IP=`echo $DOCKER_HOST|cut -d ':' -f 2|cut -d '/' -f 3`
```

You can also get the hosts' IP by running **docker-machine ls** and use any IP to make traffic call.

2. In docker swarm cluster, you can use any machine's IP to make traffic call. You can make traffic call from you local terminal.
3. Follow the steps [ML 5.2 Deployment - Amazon AWS#AmazonAWS-ServiceConfigurationData](#) for creating the API configurations.

The following is a **sample** call invocation against the MASTER_HOST_IP URL. Your service endpoint will be according to the configuration you create.

```
curl -v -H 'Host:api.example.com' 'http://MASTER_HOST_IP/v1/api/patient/1234?api_key=3bv8u3p8k5vjxmhrs6tnd6e'
```

Delete TML Cluster

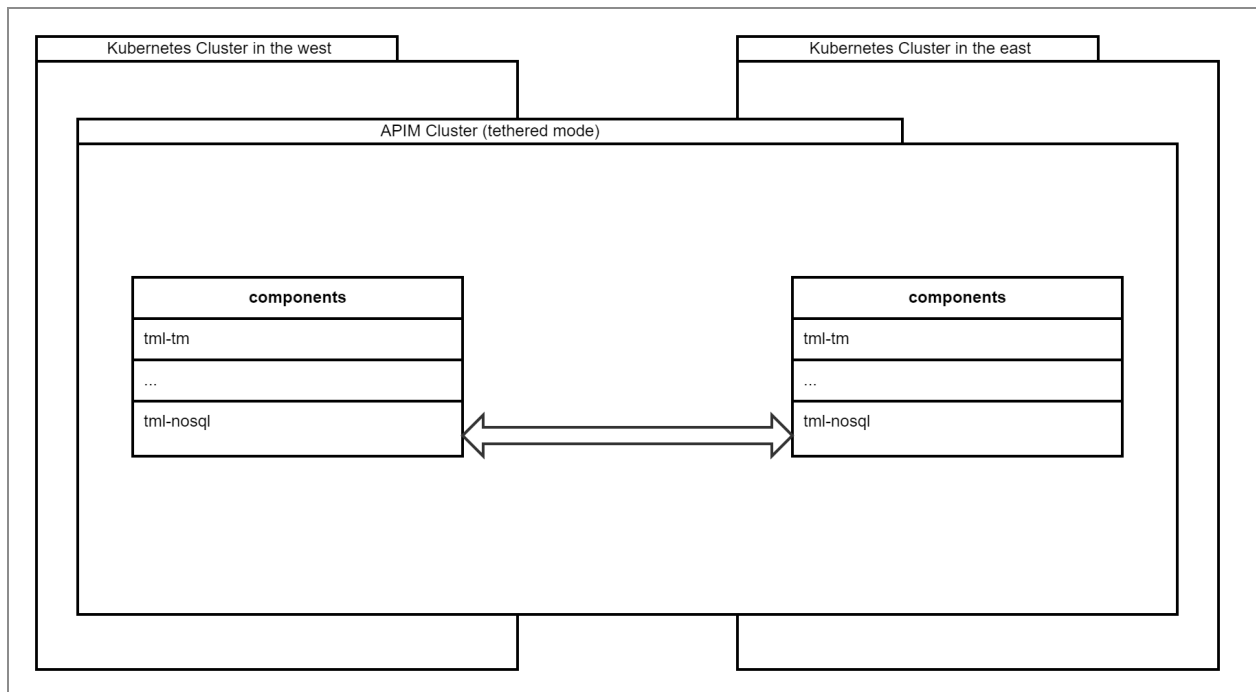
Execute the below command from inside the folder **docker-deploy/docker-swarm/manifest**

```
./delete-tml-cluster.sh
```

Deployment on one Tethered Mode Local Edition 5.5.x Cluster across Multiple Kubernetes or OpenShift Clusters

The following sections describe how to deploy one tethered mode Local Edition 5.5.x cluster across multiple Kubernetes or OpenShift clusters.

Achieving this type of deployment using Local Edition 5.5.x requires heavy customizations on deployment scripts and the tml-nosql docker image.



Scope

This section documents the steps on deploying tethered mode Local Edition 5.5.x across two Google Kubernetes Engine (GKE) clusters; deploying on other Kubernetes and OpenShift clusters only differs on externalizing tml-nosql pods.

Build Customized tml-nosql Docker Image

In Local Edition Installer, customize the function "probe_readiness" in `"/var/jenkins_home/docker-build/tmgc-nosql/install/nosql-ready-probe.sh"`:

```
probe_readiness() {
    if [ -z "$EXTERNAL_IPS" ]; then
        EXTERNAL_IP="$POD_IP"
    else
        index=$(( ${HOSTNAME: -1} + 1 ))
        EXTERNAL_IP=`echo $EXTERNAL_IPS | cut -d',' -f$index`
    fi

    ## Based on The Kubernetes Authors
    if [[ $(nodetool -h ::FFFF:127.0.0.1 status | grep $EXTERNAL_IP) == *"UN"* ]]; then
        if [[ $DEBUG ]]; then
            echo "UN";
        fi
        return 0;
    else
        if [[ $DEBUG ]]; then
            echo "Not Up";
        fi
        return 1;
    fi
}
```

In APIM Installer, customize `"/var/jenkins_home/docker-build/tmgc-nosql/install/nosql-entry-k8s.sh"`, here is the excerpt:

```
echo "EXTERNAL_IPS: $EXTERNAL_IPS"
if [ "$EXTERNAL_IPS" ]; then
    index=$(( ${HOSTNAME: -1} + 1 ))
    external_ip=`echo $EXTERNAL_IPS | cut -d',' -f$index`
    if [ "$external_ip" ]; then
        CASSANDRA_BROADCAST_ADDRESS="$external_ip"
        echo "Detected \${EXTERNAL_IPS}, broadcast address is now set to $external_ip"
    else
        >&2 echo "Detected \${EXTERNAL_IPS}, but no address was found in EXTERNAL_IPS"
```

```

at index $index"
    >&2 echo "EXTERNAL_IPS:"
    >&2 echo "-----"
    >&2 printf "$EXTERNAL_IPS\n\n"
    CASSANDRA_BROADCAST_ADDRESS=${POD_IP:-$HOSTNAME}
  fi
  else
    CASSANDRA_BROADCAST_ADDRESS=${POD_IP:-$HOSTNAME}
  fi

```

Build tml-nosql Docker Image in Local Edition Installer

Use this customized tml-nosql Docker image in deployment.

Enable Encrypted Inter-node Communication in Cassandra Cluster

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

Update "properties/tml_nosql_properties.json", 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
}

```

Generate Deployment Scripts as the Basis of Customization

This example starts with two-zone deployment manifest, all deployment scripts including customizations are on the **same** folder.

Create Kubernetes Clusters in the West and the East Region

In **terminal #1**, create Kubernetes cluster in the West region (**create-k8s-cluster-west.sh**):

```
export KUBECONFIG=${PWD}\kube-config.west

gcloud container clusters create yang-tml-cluster-multi-regions-west \
--cluster-version 1.21 \
--machine-type n1-standard-4 \
--image-type COS \
--num-nodes 3 \
--zone us-west1-a \
--node-locations us-west1-a
```

In terminal #2, create Kubernetes cluster in the East region (**create-k8s-cluster-east.sh**):

```
export KUBECONFIG=${PWD}\kube-config.east

gcloud container clusters create yang-tml-cluster-multi-regions-east \
--cluster-version 1.21 \
--machine-type n1-standard-4 \
--image-type COS \
--num-nodes 3 \
--zone us-east1-b \
--node-locations us-east1-b
```

Create Public IPs for Externalizing tml-nosql Pods

In this example, load balancers are used to externalize tml-nosql pods, each tml-nosql pod is externalized using one load balancer.

In terminal #1, run "externalize-cass-svc-0.sh", example output of "kubectl get svc":

```
cass-set-0-0-svc LoadBalancer 10.x.x.x 35.x.x.x 7000:32315/TCP 29h
cass-set-0-1-svc LoadBalancer 10.x.x.x 34.x.x.x 7000:30339/TCP 29h
cass-set-0-2-svc LoadBalancer 10.x.x.x 34.x.x.x 7000:32091/TCP 29h
```

In terminal #2, run "externalize-cass-svc-1.sh", example output of "kubectl get svc":

```
cass-set-1-0-svc LoadBalancer 10.x.x.x 34.x.x.x 7000:31657/TCP 29h
cass-set-1-1-svc LoadBalancer 10.x.x.x 35.x.x.x 7000:32634/TCP 29h
cass-set-1-2-svc LoadBalancer 10.x.x.x 34.x.x.x 7000:31324/TCP 29h
```

Example Kubernetes service yaml (**cass-set-0-0-svc.yaml**)

```

apiVersion: v1
  kind: Service
  metadata:
    labels:
      app: cass-set-0-0-svc
      name: cass-set-0-0-svc
  spec:
    type: LoadBalancer
    #loadBalancerIP: 10.xx.xx.xx
    ports:
      - name: inter-node
        port: 7001
        protocol: TCP
        targetPort: 7001
    publishNotReadyAddresses: true
    selector:
      statefulset.kubernetes.io/pod-name: cass-set-0-0

```

Notes:

- Port "7001" is for Cassandra SSL inter-node cluster communication.
- For on-premises OpenShift deployments, work with your IT department to externalize tml-nosql pods.

Deploy tml-nosql in the West and East Regions

Customize Kubernetes service for tml-nosql:

1. "cass-svc-0.yaml" defines the Kubernetes service for tml-nosql in the West region:

```

apiVersion: v1
  kind: Service
  metadata:
    labels:
      app: cass-svc
      name: cass-svc-0
  spec:
    clusterIP: None
    ports:
      - port: 9042
    selector:
      interface: cass-svc-0
  status: {}

```

2. Run **"set-cass-svc-0.sh"** to create the Kubernetes service for tml-nosql in the West region.
3. **"cass-svc-1.yaml"** defines the Kubernetes service for tml-nosql in the East region:

```
apiVersion: v1
  kind: Service
  metadata:
    labels:
      app: cass-svc
      name: cass-svc-1
  spec:
    clusterIP: None
    ports:
      - port: 9042
    selector:
      interface: cass-svc-1
  status: {}
```

4. Run **"set-cass-svc-1.sh"** to create the Kubernetes service for tml-nosql in the East region.

Note: Only port "9042" needs to be open.

Customize tml-nosql pod yaml:

Procedure"nosql-pod-0.yaml" is for tml-nosql pod in the West region, note the following changes in **env**:

```
- name: CASSANDRA_SEEDS
  value: 35.xxx.xxx.xxx,34.xx.xx.xx
- name: EXTERNAL_IPS
  value: 34.xx.xx.xx,35.xxx.xxx.xxx,34.xxx.xxx.xxx
```

Note:

- The public IP for the first tml-nosql pod in each region is selected as seed node in Cassandra.
 - "EXTERNAL_IPS" lists the public IPs for tml-nosql pods in the region in ordinal order.
1. Run **"deploy-nosql-pod-0.sh"** to deploy tml-nosql pods in the West region.
 2. **"nosql-pod-1.yaml"** is for tml-nosql pod in the East region, note the following changes in **env**:

```
- name: CASSANDRA_SEEDS
  value: 35.xxx.xxx.xxx,34.xx.xx.xx
- name: EXTERNAL_IPS
  value: 34.xx.xx.xx,35.xxx.xxx.xxx,34.xxx.xxx.xxx
```

3. Run **"deploy-nosql-pod-1.sh"** to deploy tml-nosql pods in the East region.

Deploy tml-nosql pods in the west region

In terminal #1, run **"create-tml-cluster-0-0.sh"**, please refer to the script contents for details.

Deploy tml-nosql Pods in the East Region

In terminal #2, run **"create-tml-cluster-1-0.sh"** to deploy tml-nosql pods in the East region.

Refer to the script contents for details.

Deploy Other Pods in the West Region

In terminal #1, run **"create-tml-cluster-0-1.sh"** to deploy other pods in the West region.

Refer to the script contents for details.

Deploy Other Pods in the East Region

In terminal #2, run **"create-tml-cluster-1-1.sh"** to deploy other pods in the East region.

Refer to the script contents for details.

Configuration Manager

Configuration Manager is bundled inside of the Local Edition deployment. Configuration Manager runs and exists within the cluster itself. This means you will never need to manually download the binary and run or install it on your computer. You will also never use a data.zip to import or export configuration information. The mlconfig binary talks directly to the V3 API inside of the cluster, so data state changes are reflective of the actual state of the cluster and immediate.

To use Configuration Manager, login to the cluster IP address/hostname for the configuration manager at an HTTPS port with the following URL format:

`https://<admin-configuration-url>:8443/admin`



Note: The port is also determined by the configuration as well, and may be the standard 443 if the load balancer for the deployment is configured as such.

Once there, log in and use Configuration Manager just like any other web application, (for example, the current Boomi Cloud API Management Control Center dashboard).

Access to error or usage log data is available in the form of raw (mashery_proxy_v2 format) or enriched/ECLS-formatted access logs using a local API. Configuration Manager also provides the ability to monitor system operations as healthy/normal using Local Edition data feed. You also have the ability to build diagnostic packages to facilitate assistance from Boomi Support.

Managing Configuration Data

You can manage the following configuration data:

- **Zone Settings:** Includes administrator information, and provides master operational settings for the Local Edition configuration, with fields and values that impact the entire area.
- **API Definitions:** An API definition is a set of configurable properties that specify the

function of an API within the Traffic Manager. After creating API definitions, you can create endpoints and method definitions for the API. An endpoint is a mapped location of a public host name/request path and the origin host server/endpoint path. In other words, it can be both the URI that is accessed by developers and the URI used by the Traffic Manager to send API calls to. Each endpoint contains a unique combination of request and endpoint paths, protocols, rules, accessible methods, and other settings that differentiate one endpoint from the other within an API.

- **API Packages:** Create API access Plans that can be tailored to groups of developers or partners, without any coding. Each API Plan constitutes a set of entitlement rules, limits, and filters that might be enforced when developers make calls to specific API methods or resources. An administrator can group related Plans into a Package, similar to how your cell phone or cable provider might bundle their services, thereby enabling easy management of Package Keys, including simple upgrading or downgrading of service. An API Package is a fast, easy, and efficient mode of offering APIs as a product: bundling together multiple services, endpoints, and methods, as well as restricting access down to the individual call and data elements.
- **Applications:** An application captures information about the intended use of the API by the developers. The **Applications** page displays all the registered applications. A Package Key is given to a developer to access a set of services, endpoints, and methods as defined in the respective Package Plan. A Plan associated with a Package Key can be replaced with another Plan in the same Package without changing the key string and without requiring a developer to change the application code. Configuration Manager also allows you to [Search Package Keys](#).
- **API Domains:** Manage whitelisted domain names.
- **Members:** Create, manage and delete members.
- **Roles:** Manage roles used for access control on other resources.
- **API HTTPS Client Profiles:** Manage profiles for HTTPS client authentication.
- **Content Management System:** Create, modify, and manage your portal content.
- **Interactive Documentation:** Interactive documentation for developer portal users.
- **Portal Domains:** Manage developer portal domains.
- **Portal Configuration:** Manage portal settings.

- **Organizations:** Build hierarchies to reflect your own organizational structure; assign portal content and API assets to cleanly separate responsibilities across teams.
- **Scheduled Maintenance Events:** Plan and schedule maintenance events.

Configuring Zone Settings

The **Zone Settings** page displays editable administrator information, and provides master operational settings for the Local Edition configuration, with fields and values that impact the entire area.

Home / Zone Settings

Zone Settings

Master Operational Settings

Zone Qps Limit

Call-per-second limit for all services in this area (0 to disable).

Failsafe Timeout Count

Number of times the proxy must timeout (connect or response) to trigger the failsafe.

Failsafe Timeout Period

Length of the period (in seconds) during which target host connection failures are counted.

Login Button Text

The text on the login button.

JS And CSS Configuration

Inline JS In Header

Area ID

2222

Area UUID

41a96952-ab9f-4abe-a2c8-e096c63324db

The following table describes the default fields on **Zone Settings** page.

Field	Description
Area ID	Name of Traffic Manager Domain. Needed when publishing an endpoint to Boomi Cloud API Management using Boomi Cloud Integration.
Area UUID	Universally unique identifier. Needed when publishing an endpoint to Boomi Cloud API Management using Boomi Cloud Integration.
Master Operational Settings	• Zone QPS Limit: Call-per-second limit for all services in this area. Use 0 to disable the setting. • Failsafe Timeout Count • Failsafe Timeout Period

Field	Description
Portal Settings	The Content Security Policy helps mitigate certain types of Cross Site Scripting (XSS) attacks. The default policy, listed below, is the required minimum set of entries for the portal to function properly. New entries can be added, but if there is a conflict with a default entry, the default will be used. If you use the Captcha functionality for registration, add the example Google entry provided.

Content Security Policy Editor

Content Security Policy (CSP) is an added layer of security that helps to detect and mitigate certain kinds of code injection attacks, such as Cross Site Scripting (XSS).

The **Content Security Policy** editor allows you to provide a secure Developer Portal experience for your customers - with tight control over the content the portal will attempt to consume/integrate. The new gold standard in these regards is "maintaining a robust Content-Security-Policy on your web/application server". Carefully crafting the contract for site content sources serves as a bulwark against XSS attacks.

Portal Settings

Content Security Policy

The Content Security Policy helps mitigate certain types of Cross Site Scripting (XSS) attacks. The default policy, prepopulated in the editor, is the required minimum set of entries for the portal to function properly. New entries can be added, but in the case that there is a conflict with a default, the default will be used.

Default
Script
Image
Font
Style
Connect
Object
Child
Frame
Media

☐ None
☐ All
☐ Self
☐ Data
☐ Unsafe-Inline
☐ Unsafe-Eval
☐ Unsafe-Hashes
Hosts

Space separated, * for wildcard subdomains, you can specify just a scheme e.g. https:

Press TAB to validate

Header Preview

```
script-src https://www.google.com; frame-src https://www.google.com
```

Save
Cancel

You will notice, as you begin implementation/customization of your Developer Portal, that the out-of-the-box default policy is very restrictive. This is by design - we want your server to start out with least-privilege for the default behaviors of the core application. When you need to relax a CSP directive, navigate to the Content-Security-Policy editor:

1. Login to **Configuration Manager** and click on **Zone Settings**
2. Scroll down to the **Portal Settings** section and look at your default CSP in the **Preview** pane.

The screenshot shows the 'Portal Settings' section with a 'Content Security Policy' sub-section. It includes a list of directives on the left (Default, Script, Image, Font, Style, Connect, Object, Child, Frame, Media) and a list of policy options on the right (None, All, Self, Data, Unsafe-Inline, Unsafe-Eval, Unsafe-Hashes). Below these is a 'Hosts' input field with a placeholder text: 'Space separated, * for wildcard subdomains, you can specify just a scheme e.g. https:'. A 'Header Preview' section shows the resulting CSP string: 'script-src https://www.google.com; frame-src https://www.google.com'. At the bottom are 'Save' and 'Cancel' buttons.

Default Policy

We ship a very restrictive default policy - only allowing specific assets that we need to drive our own features (example: the opt-in ReCaptcha feature, for quality-gating user registration). As a matter of fact, this 'baseline' policy will always be present in the CSP, even if you choose to completely delete it from your **Zone Settings**. You don't ever have to worry about your changes impacting our functionality, as the backend will ensure the necessary CSP allowances are in-place.

Refinement

The [Local Edition Developer Portal](#) is a functional, customizable blank-slate for you to foster compelling developer community engagement. As your Developer Portal evolves,

you will need to incorporate your brand imagery and/or other assets (potentially hosted elsewhere, from CDNs, or from inside the portal via the [ZIP File Asset Ingestion](#) mechanism). You can make any change you want to refine your portal's CSP policy in an intuitive interface.

Directive Support

Currently you have open-ended built-in support for the following ten directives:

default-src	script-src	img-src	font-src	style-src

connect-src	object-src	child-src	frame-src	media-src

For every directive above, you can:

- use check boxes to designate boolean yes/no policy elements
- add hosts to the directive
- remove hosts from the directive
- review your configured CSP
- live-preview the unsaved/proposed changes to your CSP

Adding Hosts

To add a host/domain to a directive:

1. Type into the **Hosts** editor.

Portal Settings

Content Security Policy

The Content Security Policy helps mitigate certain types of Cross Site Scripting (XSS) attacks. The default policy, prepopulated in the editor, is the required minimum set of entries for the portal to function properly. New entries can be added, but in the case that there is a conflict with a default, the default will be used.

Default	☐ None
Script	☐ All
Image	☐ Self
Font	☐ Data
Style	☐ Unsafe-Inline
Connect	☐ Unsafe-Eval
Object	☐ Unsafe-Hashes
Child	
Frame	
Media	

Hosts

Press TAB to validate

Space separated, * for wildcard subdomains, you can specify just a scheme e.g. https:

Header Preview

Save
Cancel

2. Then press **TAB**.



Note: At this point, the **Preview** will reflect your current in-page configuration, but it won't be enforced by the Developer Portal until you click **Save**.

3. Click **Save**.

SAML Metadata
[metadata](#)

Metadata for SAML identity provider.

Portal Settings

Content Security Policy

The Content Security Policy helps mitigate certain types of Cross Site Scripting (XSS) attacks. The default policy, prepopulated in the editor, is the required minimum set of entries for the portal to function properly. New entries can be added, but in the case that there is a conflict with a default, the default will be used.

Default	☐ None
Script	☐ All
Image	☒ Self
Font	☐ Data
Style	☐ Unsafe-Inline
Connect	☒ Unsafe-Eval
Object	☐ Unsafe-Hashes
Child	
Frame	
Media	

Hosts

Space separated, * for wildcard subdomains, you can specify just a scheme e.g. https:

Press TAB to validate

Header Preview

```
default-src 'self' 'unsafe-eval' https://*.mashery.com;script-src https://www.google.com;
```

Removing Hosts

When you want to remove a host from a particular CSP directive, you can simply hover your mouse over the host that you no longer want in the policy - the tag-like bubble will expand and you can click the X on the right side of the host name to remove it from the

CSP.

Portal Settings

Content Security Policy

The Content Security Policy helps mitigate certain types of Cross Site Scripting (XSS) attacks. The default policy, prepopulated in the editor, is the required minimum set of entries for the portal to function properly. New entries can be added, but in the case that there is a conflict with a default, the default will be used.

Default

Script

Image

Font

Style

Connect

Object

Child

Frame

Media

☐ None
☐ All
☒ Self
☐ Data
☐ Unsafe-Inline
☒ Unsafe-Eval
☐ Unsafe-Hashes

Hosts

Space separated, * for wildcard subdomains, you can specify just a scheme e.g. https:

Press TAB to validate

Header Preview

```
default-src 'self' 'unsafe-eval' https://*.mashery.com wikimediafoundation.org;script-src https://www.google.com;
```

Preview

As you refine your CSP, you will see that the **Preview** pane gets automatically updated to show you what the new policy will look like in the browser.

Portal Settings

Content Security Policy

The Content Security Policy helps mitigate certain types of Cross Site Scripting (XSS) attacks. The default policy, prepopulated in the editor, is the required minimum set of entries for the portal to function properly. New entries can be added, but in the case that there is a conflict with a default, the default will be used.

Default

Script

Image

Font

Style

Connect

Object

Child

Frame

Media

☐ None
☐ All
☒ Self
☐ Data
☐ Unsafe-Inline
☒ Unsafe-Eval
☐ Unsafe-Hashes

Hosts

Space separated, * for wildcard subdomains, you can specify just a scheme e.g. https:

Press TAB to validate

Header Preview

```
default-src 'self' 'unsafe-eval' https://*.mashery.com wikimediafoundation.org;script-src https://www.google.com;
```

Testing

When you want to test your policy, simply navigate to any Developer Portal page and you can inspect the Content-Security-Policy header that applies to the requests to the portal domain.

1. Click the [Content Management System Preview](#) button on a content object..
2. Open the Network debugging tool for your browser and select a call to your portal domain..

You'll find that all Developer Portal pages coming from your cluster will have your chosen policy enforced.

Configuring API Definitions and Endpoints

The **API Definitions** page displays a navigable list of API definitions. To add a new API Definition, click **New**.

Home / API Definitions							
API Definitions				my search terms	Search	All Fields ▾	Page 1 ▾ of 1 + New
Name	Version	API Service Key	Organization	Description	Created	Updated	Actions
HelloApi		n5a82wp68q4djnmvmvhwssra			Nov 16 2021 15:37	Nov 16 2021 15:37	 

The following table describes the fields on the **API Definitions** page.

Field	Description
Name	Name of the API.
Version	Version of the API.
Description	Description of the API.
Created	The date and time when the API definition was created.
Updated	The date and time when the API definition was updated.

Field	Description
Actions	
	Navigates to the API Definition Settings page, where properties of the API can be edited.
	Deletes the API definition.

The **API Definitions** page has the following sub-pages with further fields and settings that can be configured.

API Definition Settings

General settings such as API description, API version, cross-domain and robots policies can be defined on the **API Definition Settings** page.

API Definition
Security
Performance Acceleration
Portal Access Groups
Endpoints
Error Sets

Home / API Definitions / HelloApi

HelloApi

Delete

Save Cancel

API Service Key
n5a82ep68q4djnmvhwzwsra

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
HelloApi

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

Modify the description of the API. You can enter purpose, limitations, or other important information about the API.

Version

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 "http://www.macromedia.com/xml/dtds/cross-

Enter the data for cross-domain policy for API that can be accessed by using Flash.

Robots Policy

Enter the data for robots policy. If no policy information is entered, the default policy allows bots an unrestricted access to the API.

☐ RFC 3986 Encoded

Select Enabled to activate the RFC 3986 specification. By default, RFC 1738 specification is used by Mashery. You can enable the RFC 3986 specification for more flexible URLs.

After entering the policy and other details, click **Save** to save the changes.

API Definition Security

API Definition

Security

Performance Acceleration

Portal Access Groups

Endpoints

Error Sets

Home / API Definitions / HelloApi / Security

Security

Save

Cancel

☒ Enable OAuth2

Limit API calls per access token separately from API key rate limits. If enabled, each Access Token will be given its own set of limits, independent of the developer API key limits.

Grant Types

Authorization Code

Implicit

Client Credentials

Password

Select the grant types you support.

☐ Enable Token Based Rate Limits

Limit API calls per access token separately from API key rate limits.

Calls Per Second

0

Enter a throttle limit for the access tokens. Throttle is the limit of number of calls per second. Enter a throttle limit for the access tokens. Throttle is the limit of number of calls per second.

Rate Limit Ceiling

0

Enter a quota limit for the access tokens. The quota limit is applied to all access tokens granted for the API.

☐ Enable TTL

Enable the Time to Live (TTL) of OAuth Access Tokens. If set, the Access Token will expire after the specified time has passed.

Access Token TTL

0

Enter the access token TTL limit in seconds.

Access Token Type

Choose

Specify the type of access token to be generated, one of 'Bearer' or 'MAC'.

MAC Algorithm

Choose

Select one of the two algorithms to be used while securing access the tokens.

☐ Enable Refresh Token

Select Enabled to allow developers to refresh a token.

☐ Enable Refresh Token TTL

Set time to live for tokens; once TTL is reached, a refresh of the token is required.

Refresh Token TTL

0

Enter a refresh token TTL limit in seconds. The refresh token is automatically expired after it crosses the TTL value.

☐ Enable Mashery Token API

Specify whether or not Mashery will serve token requests. This requires a token endpoint being setup.

The API Definition Security page allows you to enable OAuth 2.0 for an API. The OAuth settings are applied to all endpoints that are configured to use OAuth as the security mechanism. Some of these settings can be overridden at the endpoint level.

i

Note:

This section is applicable only for those customers who are using the OAuth 2.0 Accelerator feature as part of their contract. Contact Boomi Support if you have questions about the OAuth 2.0 Accelerator.

API Definition Performance Acceleration

An API's performance can be improved by caching the responses in the cache.

API Definition

Security

Performance Acceleration

Portal Access Groups

Endpoints

Error Sets

Home / API Definitions / HelloApi / Performance Acceleration

Performance Acceleration

Save

Cancel

Cache Time-To-Live (TTL)

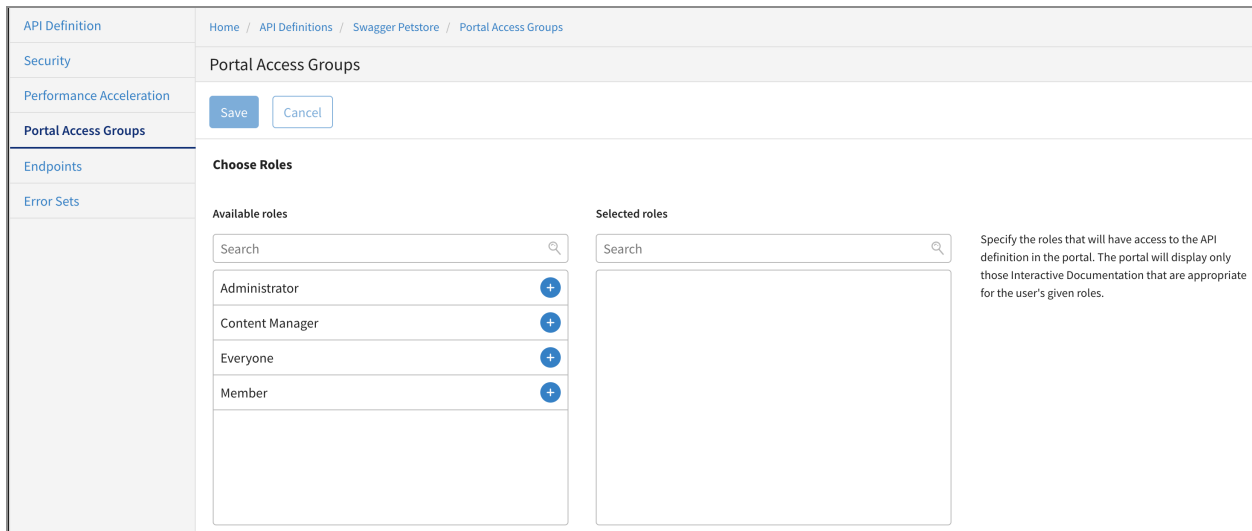
0

Control how long data is kept in the cache, with time specified in minutes.

The following table describes the field on the API Definition Performance Acceleration page.

Field	Description
Cache Time-To-Live (TTL)	The length of time the data is stored in cache. You can use the drop-down to select a TTL value. By default, the TTL is defined in minutes.

API Definition Roles



API Definition Roles configuration page. The page shows a sidebar with navigation links: API Definition, Security, Performance Acceleration, Portal Access Groups (selected), Endpoints, and Error Sets. The main content area is titled 'Portal Access Groups' and includes 'Save' and 'Cancel' buttons. Below this is a 'Choose Roles' section with two columns: 'Available roles' and 'Selected roles'. The 'Available roles' column lists Administrator, Content Manager, Everyone, and Member, each with a plus icon. The 'Selected roles' column is empty. A text box on the right explains: 'Specify the roles that will have access to the API definition in the portal. The portal will display only those Interactive Documentation that are appropriate for the user's given roles.'

In the **API Definition Roles** page, specify the roles that will have to the API definition in the Developer Portal. The portal will display only the Interactive Documentation that is appropriate for the user's given role.

API Definition Endpoints

After an API is created, you can create, modify, and delete endpoints associated with the API on the **API Definition Endpoints** page.

API Definition	Home / API Definitions / Swagger Petstore / Endpoints			
Security	Endpoints			
Performance Acceleration	my search terms Search Page 1 of 1 + New			
Portal Access Groups				
Endpoints	Name	Updated	URL	Actions
Error Sets	Swagger Petstore.user(username)	Nov 16 2021 18:23	http://acme.com/v2/user/{username}	 
	Swagger Petstore.pet(petId).uploadImage	Nov 16 2021 18:23	http://acme.com/v2/pet/{petId}/uploadImage	 
	Swagger Petstore.store.order	Nov 16 2021 18:23	http://acme.com/v2/store/order	 
	Swagger Petstore.pet(petId)	Nov 16 2021 18:23	http://acme.com/v2/pet/{petId}	 
	Swagger Petstore.user.logout	Nov 16 2021 18:23	http://acme.com/v2/user/logout	 
	Swagger Petstore.user	Nov 16 2021 18:23	http://acme.com/v2/user	 
	Swagger Petstore.pet.findByTags	Nov 16 2021 18:23	http://acme.com/v2/pet/findByTags	 
	Swagger Petstore.store.order(orderId)	Nov 16 2021 18:23	http://acme.com/v2/store/order/{orderId}	 
	Swagger Petstore.store.inventory	Nov 16 2021 18:23	http://acme.com/v2/store/inventory	 
	Swagger Petstore.user.login	Nov 16 2021 18:23	http://acme.com/v2/user/login	 
	Swagger Petstore.user.createWithList	Nov 16 2021 18:23	http://acme.com/v2/user/createWithList	 
	Swagger Petstore.pet.findByStatus	Nov 16 2021 18:23	http://acme.com/v2/pet/findByStatus	 
	Swagger Petstore.user.createWithArray	Nov 16 2021 18:23	http://acme.com/v2/user/createWithArray	 
	Swagger Petstore.pet	Nov 16 2021 18:23	http://acme.com/v2/pet	 

While you are viewing or creating endpoints, you can access the API Definition-related endpoint settings by using the navigational tabs on the left pane of the page. The following sub-pages are available from the **API Definition Endpoints** page.

Endpoints Page

API Definition	Home / API Definitions / API Definition / Endpoints / New			
Security	Endpoint			
Performance Acceleration	Save Cancel			
Portal Access Groups				
Endpoints				
Error Sets				

Name

MTLS Endpoint

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

☒ **Require Mutual TLS (MTLS)**

Published Endpoint

Protocol Domain Path

HTTPS api.acme.com

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

HTTP

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

None

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

The **Endpoint** sub-page allows you to create the name of the endpoint, as well as configure Mutual TLS (MTLS), HTTP Client Profile, Public Endpoint Address, and Boomi Cloud API Management Endpoint Address.

Configuring an Endpoint to Use MTLS

To configure an endpoint to use MTLS, select the **Require Mutual TLS** checkbox.

Protocol and Authentication Page

The **Protocol and Authentication** page allows you set the key and method details related to an endpoint. You can select parameters such as the endpoint type, supported HTTP methods, method location, and Developer API key location.

API Definition	Home / API Definitions / Swagger Petstore / Endpoints / Swagger Petstore.Store.Inventory / Protocol & Authentication	
Security	Protocol & Authentication	
Performance Acceleration	Save Cancel	
Endpoints		
Endpoint		
Protocol & Authentication		
Security		
CORS		
Call Transformation		
Performance Acceleration		
More		
Methods		
Errors		
	Endpoint Type ☒ REST ☐ SOAP ☐ JSON-RPC ☐ XML-RPC ☐ Other	Select the endpoint type. Some types might have restricted options on the rest of this page, e.g. if you select XML-RPC, then the only supported HTTP method option will be POST.
	Methods ☒ GET ☐ POST ☐ PUT ☐ DELETE ☐ HEAD ☐ OPTIONS ☐ PATCH	Select the HTTP methods that this endpoint supports. Only those selected HTTP methods will be usable by developers making calls against this Endpoint.
	Method Location ☒ Path ☐ Parameters ☐ Header ☐ Request Body ☐ Request Parameters & Body ☐ Custom	Select one or more options where the Traffic Manager will look for a method name. Method location is used in reporting.
	Method Location Identifier 	Enter the method ID. This can either be the name field that designates the method in your API calls or, in the case of a Method Location of Path, a space delimited set of numbers designating the locations in the request path that should be recorded as the method, e.g. 2 4 to specify that the method for http://someserver.com/rest/v1/products/1232/inventory is 'products inventory'.
	Request Authentication Type API key	Set how authentication will be handled for the API; this authentication is for the developer making calls against the API. If you select custom, a valid adapter name will be required. Selecting OAuth 2.0 or Custom will require further configuration.
	Developer's API Key Location ☐ Path ☐ Parameters ☐ Header ☐ Request Body ☒ Request Parameters & Body ☐ Custom	Select where the Traffic Manager must look for a developer's API key.
	Key Field Identifier api_key	Enter the API Key ID. This can either be the name field that designates the key in your API calls or, in the case of a Key Location of Path, a space delimited set of numbers designating

Security Page

The **Security** page allows you to enable System Domain Authentication.

API Definition	Home / API Definitions / Swagger Petstore / Endpoints / Swagger Petstore.Store.Inventory / Security	
Security	Security	
Performance Acceleration	Save Cancel	
Endpoints		
Endpoint	☒ System Domain Authentication Required For the connection between the Traffic Manager and the Endpoint system, specify whether or not there is authentication required.	
Protocol & Authentication		
Security		
CORS		
Call Transformation		
Performance Acceleration		
More		
Methods		
Errors		

CORS Page

The **CORS** page allows you to configure Endpoint Cross-Origin Resource Sharing, a W3C standard that defines how the browser and server should interact when cross-domain requests are made.

API Definition	Home / API Definitions / Swagger Petstore / Endpoints / Swagger Petstore.Store.Inventory / CORS	
Security	CORS	
Performance Acceleration	Save Cancel	
Endpoints		
Endpoint	☒ Enable CORS Activate/deactivate the feature.	
Protocol & Authentication	☐ Allow All Domains Specify if pre-flight requests made from any domain are passed through the Traffic Manager. When disabled, only pre-flight requests from specified domains will be allowed.	
Security		
CORS		
Call Transformation		
Performance Acceleration		
More		
Methods		
Errors		

Call Transformations Page

The **Call Transformations** page allows you to apply processing logic to an API call before it is sent to the host API servers and after the call is returned from the client API servers.

API Definition	Home / API Definitions / Swagger Petstore / Endpoints / Swagger Petstore.Store.Inventory / Call Transformation	
Security	Call Transformation	
Performance Acceleration	Save Cancel	
Endpoints		
Endpoint		
Protocol & Authentication		
Security		
CORS		
Call Transformation		
Performance Acceleration		
More		
Methods		
Errors		

Adapter

Enter name of processing adapter.

☐ **Perform Preprocess**

Select Enabled to allow pre-processing.

Preinputs

Enter the data or values that can be used by the Traffic Manager while pre-processing the calls.

☐ **Perform Postprocess**

Select Enabled to allow post-processing.

Postinputs

Enter the data or values that can be used by the Traffic Manager while post-processing the calls.

Performance Acceleration

Endpoint performance can be optimized by managing the cache. If your call responses are cacheable, specifying appropriate caching rules reduces the end-to-end latency time of API calls. The caching rules set at an endpoint-level, as shown in the following image, affect all calls made to the endpoint.

API Definition	Home / API Definitions / Swagger Petstore / Endpoints / Swagger Petstore.Store.Inventory / Performance Acceleration	
Security	Performance Acceleration	
Performance Acceleration	Save Cancel	
Endpoints		
Endpoint		
Protocol & Authentication		
Security		
CORS		
Call Transformation		
Performance Acceleration		
More		
Methods		
Errors		

Cache Time-To-Live (TTL)

Control how long data is kept in the cache, with time specified in minutes.

Honor Control Headers

☒ None ☐ Surrogate ☐ Cache

Select surrogate to activate surrogate cache control or select cache to allow responding from the cache control.

☐ **Use Developer Key In Content Cache Key**

Including the Developer Key in the Content Cache Key will result in individual developer's calls being cached. Disabling this setting means that API calls can be cached and re-used across a group of developers.

Headers To Include In Content Cache Key

The HTTP cache-control headers can be included in the responses for fine-tuned and granular caching. The cache-control header in the responses overrides the global TTL value, and sets TTL values on an object by object basis.

More Page

The **More** page allows for configuring Traffic-Manager-specific settings. You can configure the Traffic Manager to include or exclude headers from the requests and responses.

API Definition
Security
Performance Acceleration
Endpoints
Endpoint
Protocol & Authentication
Security
CORS
Call Transformation
Performance Acceleration
More
Methods
Errors

Home / API Definitions / Swagger Petstore / Endpoints / Swagger Petstore.Store.Inventory / More

More

Save Cancel

Endpoint Response Wait Time

2

Enter time (in seconds) to wait for a response from the endpoint. If no response is received until the stipulated time, Mashery throws the 504 Gateway Timeout error.

Seconds To Attempt Initial Connection

2

Enter time (in seconds) for which an attempt of initial connection to the endpoint can be made. On the failure of establishing a connection to the TCP layer of the client server, Mashery throws the 504 Gateway Timeout error.

Number Of HTTP Redirects To Follow

0

Enter a number of HTTP redirects to follow. By default, Mashery does not handle any redirects.

☐ Enable Cookies During Redirects

Select Enabled to enable cookies during redirects.

☐ Gzip Passthrough

Select Enabled to enable GZIP passthrough

☐ Force Gzip Accept-Encoding On Backend Call

Select Enabled to force GZip directive on the backend server. The Accept-encoding value passed by the client server is overridden.

☐ Remove API Key And Signature From Endpoint Call

Enabling will remove API key and signature from an endpoint call. This setting is for the calls made to the client API server by Mashery.

☐ Allow Calls Having No API Key

Select Enabled to allow calls without API keys. This setting is for the calls made to Mashery. Further configuration required in conjunction with this option.

☐ Swap Incoming API Credentials For Mashery Credentials

Select Enabled to swap incoming API credentials for Mashery credentials before passing the API calls to the client API server.

Use This API Key Instead

The developer key associated with an API call will be swapped with this key before the Traffic Manager can pass the call to the host server. Required field if Swap Incoming API credentials for Mashery Credentials is set.

Sign Endpoint Request With This Secret

API Call passed to the host server will be signed with this secret.

API History

The **API History** page displays the history of changes made for the selected API definition. In addition to viewing historical information, users have the ability to compare versions, and revert to an earlier version of an API definition.

To compare API versions, select the versions using the checkboxes, then click **Compare Selected**. A pop-up window displays and lists any changes between the selected versions.

Compare API Versions: aurora-api			
Field	Change to value	Value in Version 5	Value in Version 6
Endpoints > aurora-endpoint1 > General > Name for this Endpoint	Updated	aurora-endpoint1	aurora-endpoint

To revert to an earlier version of an API, click the **Revert** icon in the **Actions** column for the desired API version.

Enabling HSSM for an Endpoint

HSSM (Highly Secure Secret Management) involves securing credentials to protect sensitive information.

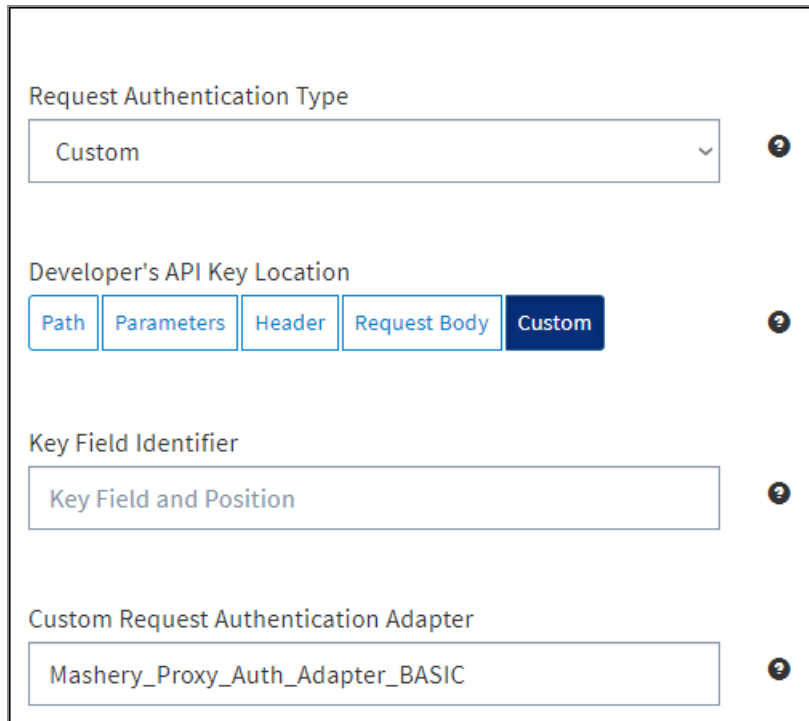
Local Edition supports the sending request with the apikey and authentication credentials in the authorization header for service endpoint that requires apikey and basic authentication. For both non-HSSM and HSSM-enabled areas, the apikey from authorization header is used to identify service endpoint of an application. For HSSM-enabled area, the apikey and secret are hashed during authentication to match the stored credentials.

Authentication is processed by an authenticator adapter and apikey is retrieved by apikey locator. Since the authenticator adapter has the same name as the apikey locator for a service endpoint, the name of the authenticator adapter is also used for the apikey locator.

As a result, the name for authenticator and apikey locator to use is based on the authenticator set for the service endpoint.

Configuration for an endpoint for a non-HSSM and HSSM-enabled area is done in the **Endpoint Key & Method Detection** page of Configuration Manager, as shown below.

Non-HSSM Area



The screenshot displays the 'Non-HSSM Area' configuration interface. It contains four main sections, each with a help icon (question mark in a circle) to its right:

- Request Authentication Type:** A dropdown menu currently set to 'Custom'.
- Developer's API Key Location:** A set of five buttons: 'Path', 'Parameters', 'Header', 'Request Body', and 'Custom'. The 'Custom' button is highlighted in dark blue, indicating it is the selected option.
- Key Field Identifier:** A text input field containing the placeholder text 'Key Field and Position'.
- Custom Request Authentication Adapter:** A text input field containing the value 'Mashery_Proxy_Auth_Adapter_BASIC'.

Set the **Custom Request Authentication Adapter** to Mashery_Proxy_Auth_Adapter_BASIC.

HSSM-Enabled Area

Request Authentication Type

Custom

?

Developer's API Key Location

Path

Parameters

Header

Request Body

Custom

?

Key Field Identifier

Key Field and Position

?

Custom Request Authentication Adapter

Mashery_Proxy_Auth_Adapter_HSSM

?

Set the **Custom Request Authentication Adapter** to `Mashery_Proxy_Auth_Adapter_HSSM`

Configuring API Packages and Plans

The **API Packages** page allows you to manage (create, view and delete) packages and plans associated with APIs.

The following table describes the fields on the **Packages** page.

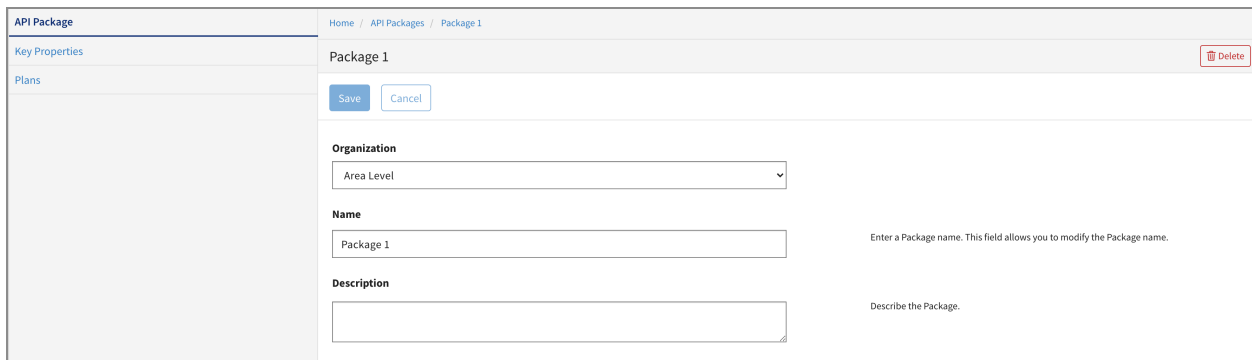
Field	Description
Name	Name of the Package.
Description	Description of the Package.
Created	The date and time when the Package was created.

Field	Description
Updated	The date and time when the Package was updated.
Actions	
	Navigates to the API Package Edit page, where properties of the Package can be edited.
	Deletes the Package.

The following sub-pages are available from the **API Packages** page.

API Package Edit Page

The **API Package Edit** page allows you to configure the name and description of the package.



API Package Key Properties Page



API Package Plans Page

The **API Package Plans** page allows you to configure the name and description of the package.

API Package	Home / API Packages / Package 1 / Plans						
Key Properties	Plans						
Plans	my search terms Search All Fields + New						
	Name	Description	Throttle	Quota	Quota Period	Created	Updated
	Plan 1		2	5000	day	Nov 16 2021 18:33	Nov 16 2021 18:33
							Actions

The following sub-pages are available from the **API Packages Plans** page.

API Package Plans Plan Page

The **API Package Plans Plan** page allows you to configure the name, description and status for a package's plan.

API Package	Home / API Packages / Package 1 / Plans / Plan 1	
Key Properties	Plan 1	
Plans	Save Cancel	
Plan		
Rate Limits		
Key Properties		
Portal Access Groups		
Definitions		
Notification Settings		
	Name Plan 1	Enter an API Plan name for future reference. API Plan name is displayed while creating Packages and generating reports.
	Description 	Describe the API Plan. Enter the purpose, limitations, or other important information about the API Plan.
	Status Active	Plan status is available for internal reference only. It does not have any effect on the Plan or related Applications and Package Keys.

API Package Plans Rate Limit Page

The **API Package Plans Rate Limit** page allows you to configure call rate limits for a particular package's plan.

API Package	Home / API Packages / Package 1 / Plans / Plan 1 / Rate Limits	
Key Properties	Rate Limits	
Plans	Save Cancel	
Plan		
Rate Limits		
Key Properties		
Portal Access Groups		
Definitions		
Notification Settings		
	Throttle 2	Enter a throttle limit for the package keys. Throttle is the limit of number of calls per second.
	☒ Allow Package Key Throttle Override	Allow the throttle (queries per second) settings for package keys to override the limits set in this plan.
	Quota 5000	Enter a number that is the upper limit of calls that a package key can make in the given time period.
	Quota Period Day	Time period used to control the maximum number of calls to be made by a given developer.
	☒ Allow Package Key Quota Override	Allow the quota settings per individual package keys to be changed to override the limits set in the plan.

API Package Plans Key Properties

The **API Package Plans Key Properties** page provides settings to help you to allow or restrict the developers or administrators from generating keys for the Package Plan.

API Package	Home / API Packages / Package 1 / Plans / Plan 1 / Key Properties	
Key Properties	Key Properties	
Plans	Save Cancel	
Plan		
Rate Limits		
Key Properties		
Portal Access Groups		
Definitions		
Notification Settings		
	☒ Self Service Key Provisioning Select to allow developers to get a key for themselves.	
	☒ Admin Key Provisioning Select to allow administrators to generate key for developers.	
	Maximum Keys Allowed 4	Enter the maximum number of keys that can be generated for a particular developer.
	Number Of Keys Before Review 	Select the number of keys that can be generated without moderation.

The Package Key is generated for the developers to access a particular Package and the associated Plan. A key adapter and other key properties can be set on the **Key Properties** page.

API Package Plans Roles

The **API Package Plans Roles** page allows you to specify the roles that will have access to the API Plan in the portal.

API Package	Home / API Packages / Package 1 / Plans / Plan 1 / Portal Access Groups	
Key Properties	Portal Access Groups	
Plans	Save Cancel	
Plan		
Rate Limits		
Key Properties		
Portal Access Groups		
Definitions		
Notification Settings		
	Choose Roles Available roles Search - Content Manager + - Everyone + - Member +	Selected roles Search Administrator - Specify the roles that will have access to the API Plan in the portal. This information is used on the Application registration pages; only those users with the right roles will be able to get keys for the API Plans.

API Package Plans Definitions

The **API Package Plans Definitions** page allows you to activate and link the API definition to a plan.

API Package	Home / API Packages / Package 1 / Plans / Plan 1 / Definitions									
Key Properties	Definitions									
Plans										
Plan										
Rate Limits										
Key Properties										
Portal Access Groups										
Definitions										
Notification Settings										
	Available API Definitions Page 1 of 1	Activated API Definitions								
	<table border="1"> <thead> <tr> <th>Name</th> <th>Actions</th> </tr> </thead> <tbody> <tr> <td>Swagger Petstore</td> <td> Activate </td> </tr> </tbody> </table>	Name	Actions	Swagger Petstore	Activate	<table border="1"> <thead> <tr> <th>Name</th> <th>Actions</th> </tr> </thead> <tbody> <tr> <td>Swagger Petstore</td> <td> Configure Delete </td> </tr> </tbody> </table>	Name	Actions	Swagger Petstore	Configure Delete
Name	Actions									
Swagger Petstore	Activate									
Name	Actions									
Swagger Petstore	Configure Delete									

The Package Key is generated for the developers to access a particular Package and the associated Plan. A key adapter and other key properties can be set on the **Key Properties** page.

Selecting Endpoints for a Definition

To activate endpoints for a definition:

1. Navigate to **API Packages > selected Package > Plans > selected Plan > Definitions > selected Definition > Endpoints**.

API Package

Key Properties

Plans

Plan

Rate Limits

Key Properties

Portal Access Groups

Definitions

Notification Settings

Home / API Packages / Package 1 / Plans / Plan 1 / Definitions / Swagger Petstore / Endpoints

Endpoints

Available Endpoints

Name	Actions
Swagger Petstore.user(username)	Activate
Swagger Petstore.pet(petId).uploadImage	Activate
Swagger Petstore.store.order	Activate
Swagger Petstore.pet(petId)	Activate
Swagger Petstore.user.logout	Activate
Swagger Petstore.user	Activate
Swagger Petstore.pet.findByTags	Activate
Swagger Petstore.store.order(orderId)	Activate
Swagger Petstore.store.inventory	Activate
Swagger Petstore.user.login	Activate
Swagger Petstore.user.createWithList	Activate
Swagger Petstore.pet.findByStatus	Activate
Swagger Petstore.user.createWithArray	Activate
Swagger Petstore.pet	Activate

Activated Endpoints

Name	Actions
Swagger Petstore.store.inventory	 
Swagger Petstore.store.order	 

2. On the **Endpoints** page, click the **Activate** link for the endpoint in the **Available Endpoints** list. The endpoint moves to the **Activated Endpoints** list.

Configuring Apps and Package Keys

The **Applications** page allows you to configure applications and manage the package keys associated with applications.

The following table describes fields displayed on the **Applications** page.

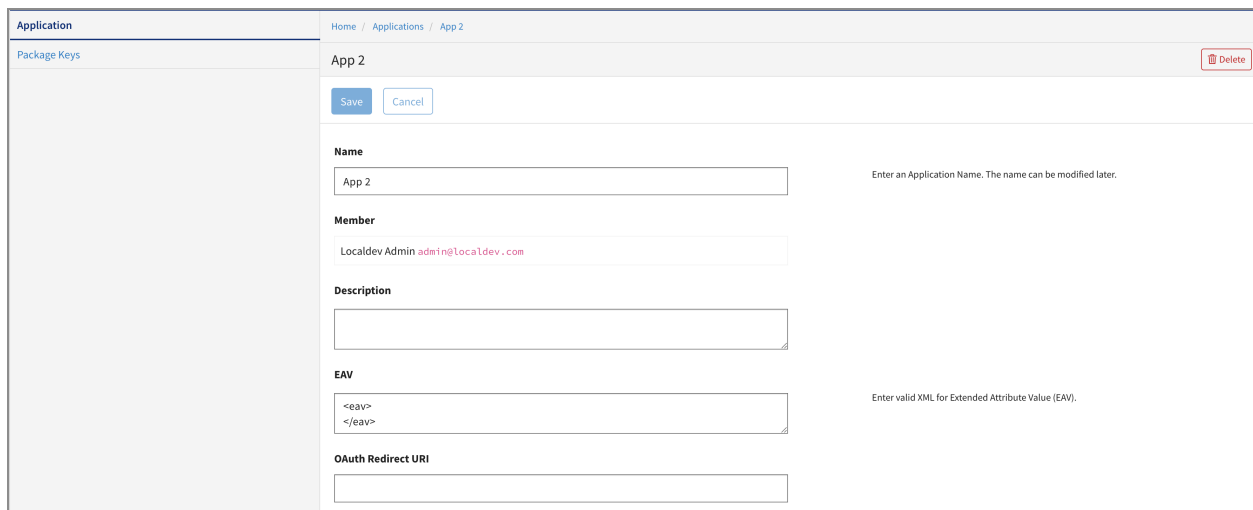
Column Name	Description
Name	The name of the application.

Column Name	Description
Description	The description of the application.
Created	The date and time when the application was created.
Updated	The date and time when the application was last updated.
Actions	
	Click to modify the application. The Applications Settings page is displayed. You can modify the fields, as required.
	Click to delete the application.

The following sub-pages are available from the **Applications** page.

Applications Settings Page

The **Applications Settings** page allows you to configure settings, such as name and description, for a specific application.



The screenshot shows the 'Applications Settings' page for 'App 2'. The page has a left sidebar with 'Application' and 'Package Keys' links. The main content area has a breadcrumb 'Home / Applications / App 2' and a 'Delete' button. Below the breadcrumb are 'Save' and 'Cancel' buttons. The form fields include:

- Name:** A text field containing 'App 2' with a placeholder 'Enter an Application Name. The name can be modified later.'
- Member:** A text field containing 'Localdev Admin admin@localdev.com'.
- Description:** An empty text area.
- EAV:** A text field containing '<eav>' and '</eav>' with a placeholder 'Enter valid XML for Extended Attribute Value (EAV)'.
- OAuth Redirect URI:** An empty text field.

Package Keys Page

The **Package Keys** page allows you to configure the package keys associated with an application.

Application	Home / Applications / App 2 / Package Keys						
Package Keys	Package Keys						
	my search terms Search + New						
API Key	Status	Package - Plan	Throttle	Quota	Quota Period	Updated	Actions
av8j32q3pqwhy75794cp4mcx	waiting	Package 1 - Plan 1	2	5000	day	Nov 16 2021 18:43	

The throttle and the quota limits can be specified at the key level. These limits override the limits set at the Plan level.

To set key overrides:

1. On the **Package Keys** page, click the specific key, (or click the edit icon for the specific key). The **Package Key** page is displayed.

Application	Home / Applications / App 2 / Package Keys / Package Key		
Package Keys	Package Key		
Package Key	Save Cancel		
	Package 1 Plan 1 API Key Secret Status Rate Limit Type Rate Limit Ceiling QPS Limit Type QPS Limit Ceiling EAV		
	av8j32q3pqwhy75794cp4mcx Waiting Default 5000 Default 2 <eav></eav>		
	When enabled, calls made from applications using this key will be processed. Enter valid XML for Extended Attribute Value (EAV).		

2. Set the rate limits and QPS limits for this key list, then click **Save**.

Configuring API Domains

The **API Domains** page allows you to manage domains. The Domain name is required to be able to enter input data for a new endpoint.

Home / API Domains	API Domains		
	my search terms Search All Fields + New		
Domain	Status	Created	Actions
acme.com	active	Nov 16 2021 18:22	
petstore.swagger.io	active	Nov 16 2021 18:23	

Adding a Domain

To add a new domain:

1. From the API Domains page, click on **New Domain**.



Home / API Domains / New

API Domain

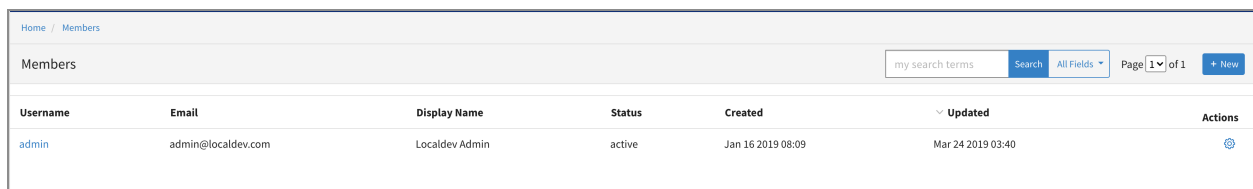
Save Cancel

Domain

2. In the **API Domain** window, enter a domain in the **Domain** field, then click **Save**.

Configuring Members

The **Members** page displays a navigable list of registered users for the Developer Portal.



Home / Members						
Members						
				my search terms	Search	All Fields
				Page	1	of 1
				+ New		
Username	Email	Display Name	Status	Created	Updated	Actions
admin	admin@localdev.com	Localdev Admin	active	Jan 16 2019 08:09	Mar 24 2019 03:40	

Adding a New Member

To add a new member to the Developer Portal:

1. Click the **New** button on the **Members** page.
2. In the **Dev Portal User** page, enter the following information:
 - Display Name
 - First Name
 - Last Name
 - Company
 - Email

- Username
- Status

Member	Home / Members / Devportal User
Portal Access Groups	Devportal User Delete
	Save Cancel Display Name Devportal user First Name Devportal Last Name user Company internal Email devportal@acme.com Username devportal Status waiting

3. Click **Save**.

Deleting a Member

To delete a member from the Developer Portal:

1. In the **Members** page, click the delete icon next to the specified member.
2. In the **Member Delete Confirmation** window, click **Yes** to delete the member.

Note: The operation cannot be undone.

Assigning Developer Portal Roles to Members

You can assign Developer Portal roles to members in an organization. Specific roles can

be added to or removed from member accounts on the **Members** page. Members will have role-based access to the Boomi Developer Portal resources.

To assign a role to a member:

1. In the **Members** page, click the **settings** icon for the selected member.
2. In the **Roles** page, select a role from the **Choose Roles** list, then click **Save**.

Configuring Roles

The **Roles** page displays Developer Portal roles.

Home / Portal Access Groups			
Portal Access Groups		my search terms Search	Page 1 of 1 + New
Name	Created	Updated	Actions
Content Manager	Nov 10 2019 23:50	Nov 10 2019 23:50	
Administrator	Jul 12 2009 13:34	Jan 15 2019 05:40	
Everyone	Jul 12 2009 13:34	Feb 12 2018 23:09	
Member	Jul 12 2009 13:34	Apr 27 2016 23:19	

The following table describes the Developer Portal roles and their permissions.

Role	Description
Custom	

Role	Description
Administrator	Has access to all screens.
Everyone	
Member	

Creating a Custom Role

To create a **Custom** role:

1. In the **Roles** page, click the **New** button.
2. In the **Custom** page, enter a name for the Custom role, then click **Save**.

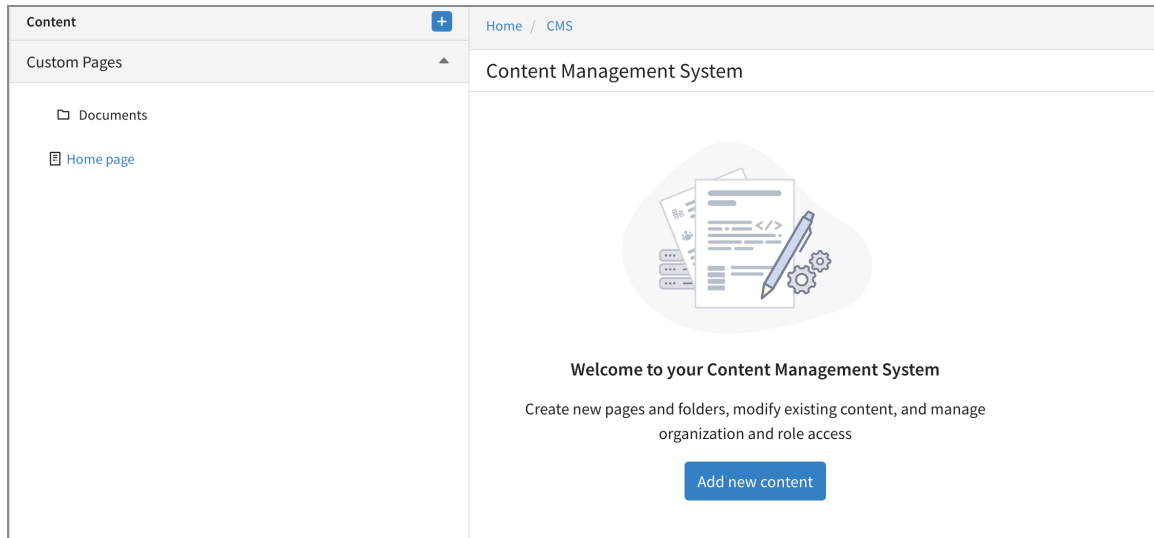
Configuring API HTTPS Client Profiles

In the **API HTTPS Client Profiles** page, keys and certificates that were configured in and managed through Local Edition can be used for HTTPS client authentication.

The **API HTTPS Client Profiles** page displays a list of HTTPS client profiles created by the Administrator for a particular customer.

Adding an API HTTPS Client Profile

1. On the **API HTTPS Clients Profile** page, click **New**.

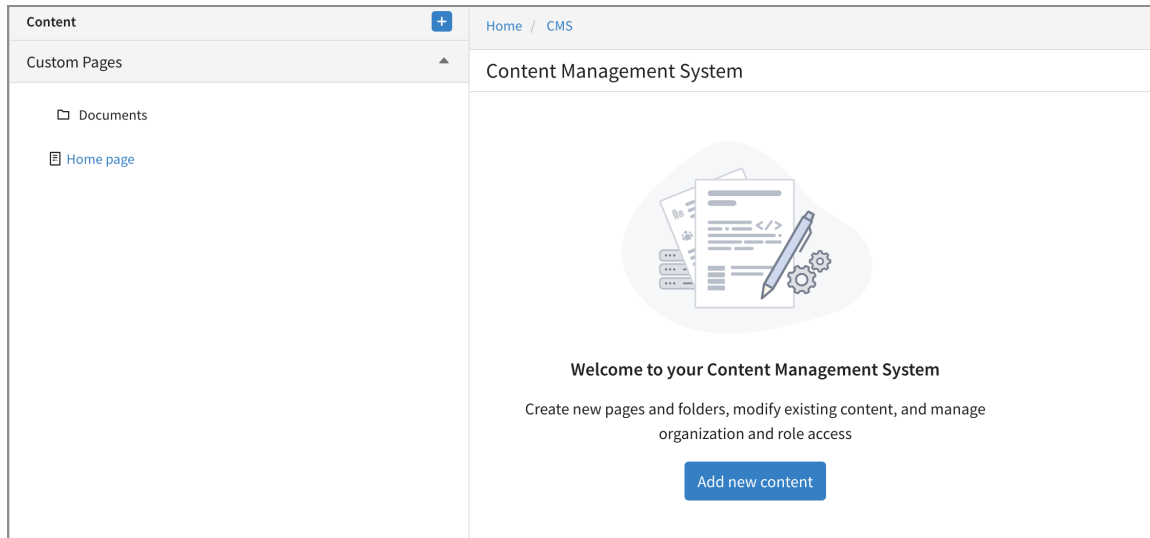


2. In the **API HTTPS Client Profile** page, enter the following information in the fields:
 - **Name** - Enter a name for the HTTPS client profile.
 - **Verify Hostname** - Enable the checkbox to have the hostname verified.
 - **API HTTPS Client Identity**- Select an identity for the HTTPS client profile.
 - **Choose API HTTPS Client Trusts** - Select a trust for the HTTPS client.
3. Click **Save**.

Configuring the Content Management System

The **Content Management System** page allows you to add, modify, or delete custom HTML pages for your Developer Portal.

The **Content Management System** page has a built-in WYSIWYG editor that is enriched with formatting tools similar to any word processing program. The **Preview** feature in Configuration Manager redirects you to that specific revision of that portal page. A separate style sheet and JavaScript can be customized and added to every page, thereby providing specific look-and-feel and even functionality. You can add role-based permissions to control who can view the pages.



Creating Custom Pages

1. On the **Content Management System** page, click **Add New Content**.
A page is added to the **Custom Pages** folder in Draft mode..
2. Specify the **Type** as **Page**.
3. Enter the page title in the **Title** field.
4. Specify a **Path** for this page.
5. Select the roles that will have access to this page in **Access Permissions** field.
6. In the **Organization** field, use the drop-down to select the **Area Level** that will have access to this page.

i Note: For more information on Organizations, see [Managing Organizations](#).

7. In the **Parent** field, use the drop-down to select the parent page.
8. In the **Order** and **Page** fields, use the drop-downs to place the page **Above** or **Below** an existing page.
9. Click **Save** to create the page.
10. In the **Body Content** field, enter the text. Using the editor, you can format the text.

i Note: In the **Advanced options** section, you can modify the following options for the page:

- URL path
- Access permissions
- Organization settings
- Page status
- Page hierarchy

For more information, see [Setting Advanced Options](#).

11. Click **Preview** to view this version of the page on the Developer Portal.

i Note: The links to **Preview** are portable, so long as the recipients are supposed to have access; otherwise, they will get a 404 error message.

12. Click **Publish** to publish the page on the Developer Portal.

Adding a Folder

To add a folder, click **New**, specify the **Type** as **Page**, then specify a **Title** for the folder.

Adding a Page to a Folder

To add a page to a folder, click the folder, then click **New**.

Setting Advanced Options

Use the **Advanced options** settings to configure the following properties for an existing CMS content page.

- URL path

▼ Advanced options

URL path

Access permissions

Organization settings

Page status

Page hierarchy

Path

welcome

- Access permissions

Advanced options

URL path

Access permissions

Organization settings

Page status

Page hierarchy

Access Permissions

Specify the roles that will have access to this page. The portal will display only those pages that are appropriate for the user's given roles.

Acme Company Administrator	
Acme Company API Manager	
Acme Company Community Manager	
Acme Company Reports User	
Administrator	
Content Manager	
Engineering Administrator	
Engineering API Manager	
Engineering Community Manager	
Engineering Reports User	
Everyone	
Member	

- Organization settings

▼ Advanced options

URL path

Access permissions

Organization settings

Page status

Page hierarchy

Organization settings

Organization

Area Level

- **Page status**

Advanced options

URL path

Access permissions

Organization settings

Page status

Page hierarchy

Page status

Modify the visibility of this page or revert to the last previously published version.

Page Visibility

☒ Visible

☐ Hidden

Page history

You can revert to the last published version of this page and make edits to that content.

Revert To Last Published

- **Page hierarchy**

Advanced options

URL path

Access permissions

Organization settings

Page status

Page hierarchy

Page hierarchy

Position where this page is located in the navigation tree.

Parent

Documents

Order

Choose

Page

Choose

Managing Developer Portal Content

On the **Content Management System** page, content placed in the **Documentation** folder will, by default, render with an automatically generated navigation menu of all of the content under that folder.

Disabling the Navigation Menu Feature

You can turn this navigation menu off by setting `docsNav` to `false` in your `configurations.js` file.

```
portal.settings({  
  docsNav: false  
});
```

Styling the Documentation Navigation Menu

The navigation menu and content are wrapped in an element with the `.docs-wrapper` class. The navigation menu itself lives inside an element with the `.docs-nav` class, while the content is in an element with the `.docs-content` class.

```
<div class="docs-wrapper">  
  <div class="docs-nav">  
    <!-- Nav menu -->  
  </div>  
  <div class="docs-content">  
    <!-- Content -->  
  </div>  
</div>
```

The navigation menu itself has a class of `.docs-nav-list`. Individual items in the navigation have a `.content-nav` class, as well as a unique ID. Items with a nested navigation below them have a `.content-nav-sublist` class.

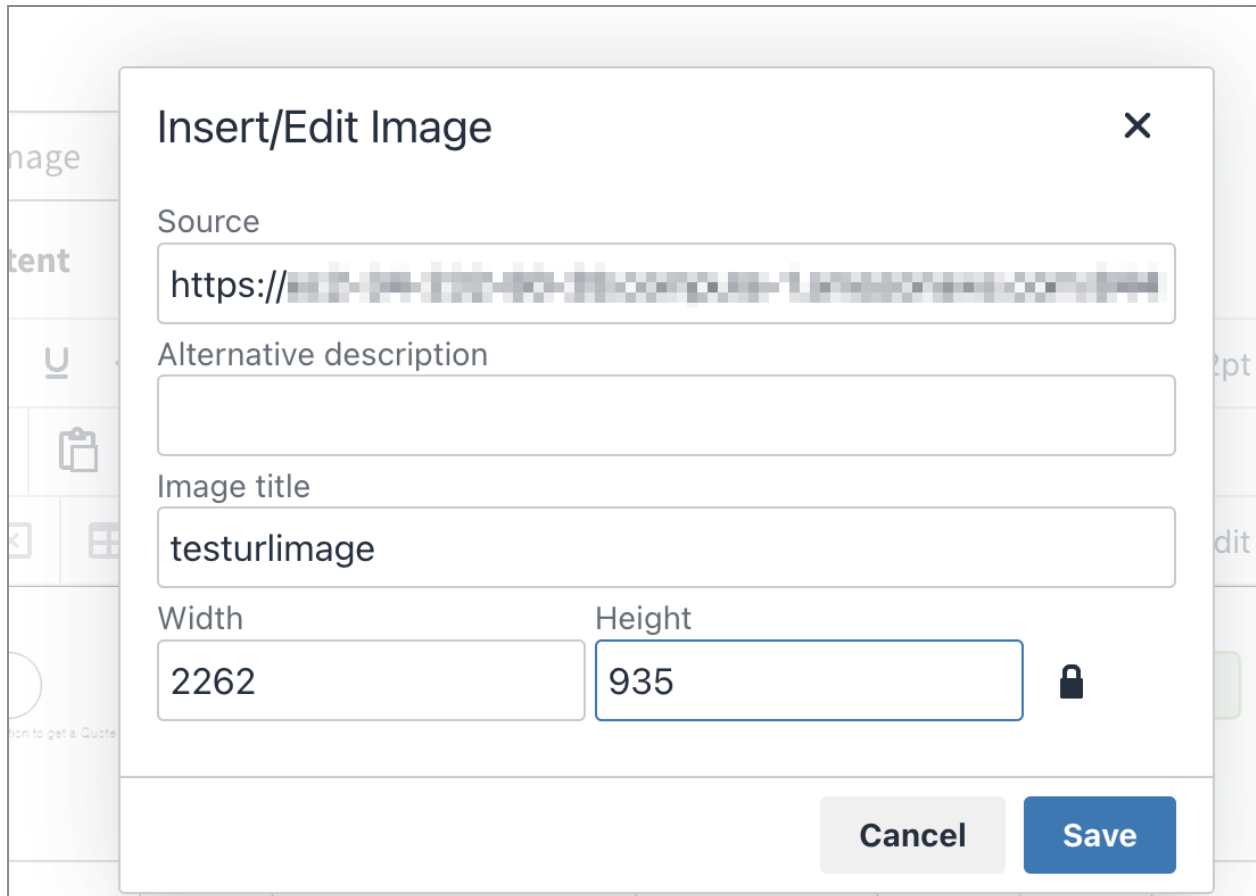
Adding a Heading

You can add a heading above the documentation navigation menu by adding a translation to your `customizations.js` file. The view is ``page``, and the id is ``navTitle``.

```
portal.addTranslation('page', 'navTitle', 'In the Docs', 'en');
```

Adding an Image

You can add an image on CMS pages by providing a complete URL (for example, `https://dev.acme.com:10443/admin/assets/illustration_cms.png`) which will be internally converted to its relative path and stored as `../../assets/illustration_cms.svg`.



The screenshot shows a modal dialog titled "Insert/Edit Image" with a close button (X) in the top right corner. The dialog contains the following fields and controls:

- Source:** A text input field containing a URL: `https://dev.acme.com:10443/admin/assets/illustration_cms.png`.
- Alternative description:** An empty text input field.
- Image title:** A text input field containing the text `testurlimage`.
- Width:** A text input field containing the value `2262`.
- Height:** A text input field containing the value `935`. To the right of this field is a lock icon.
- Buttons:** At the bottom right, there are two buttons: "Cancel" (light gray) and "Save" (blue).

i Note: Note: You must add the complete URL and not the relative path.

Searching Package Keys

The **Search Package Keys** page allows you to search package keys by API Key. Enter an API key in the search field to search package keys.

Home / Package Keys

API Key
abcd1234

Plan Name
existing plan

API Package Name
existing package

Application Name
existing app

Username
username

Search Clear

Page 1 of 1

API Key	Status	Application Name	Package - Plan	Throttle	Quota	Quota Period	Updated	Actions
av8j32q3pqwhy75794cp4mcx	waiting	App 2	Package 1 - Plan 1	2	5000	day	Nov 16 2021 18:43	

Configuring Interactive Documentation

Interactive Documentation is an interactive documentation system for RESTful and SOAP-based APIs. You can specify Interactive Documentation page settings and add definitions to selected APIs. The **Interactive Documentation** page lists the Interactive Documentation that is registered on the Developer Portal.

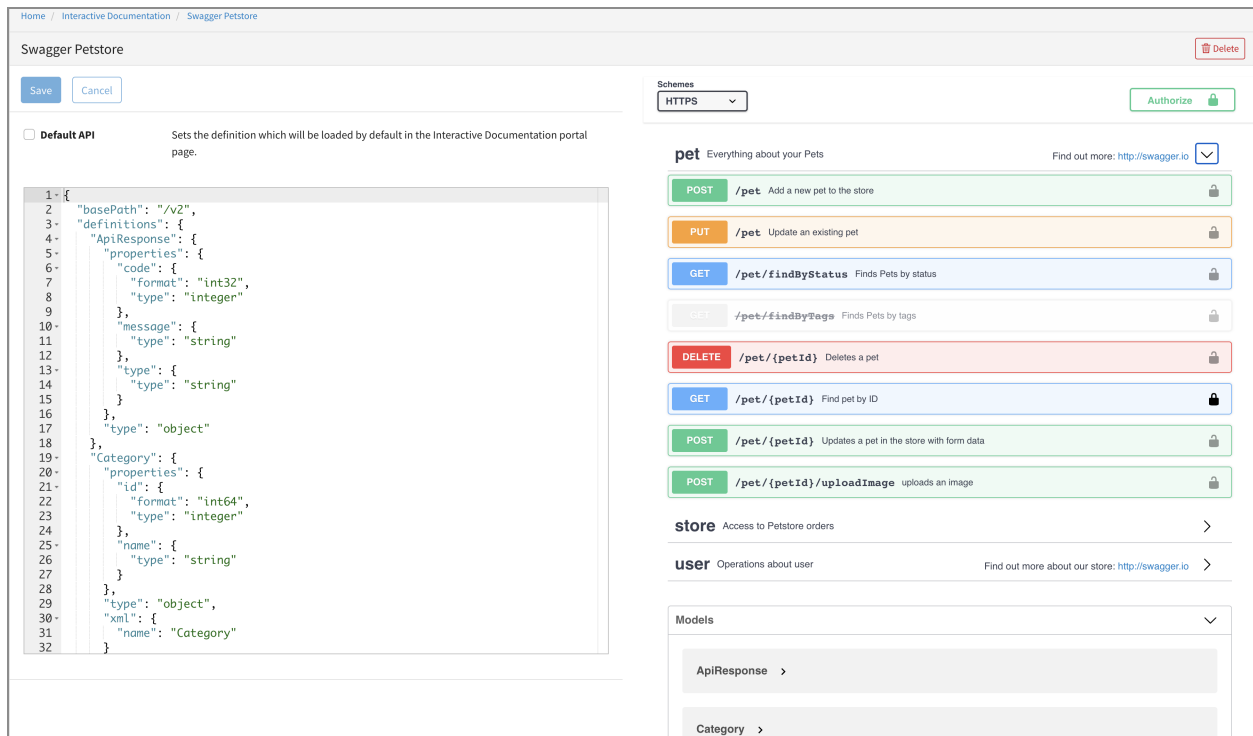
Adding Interactive Documentation

1. From the **Interactive Documentation** page, click the **New** button. This launches a **New Interactive Documentation** window.
2. Select an API Definition from the drop-down list.
3. Specify the Interactive Documentation as a Swagger document.

Note: This setting cannot be changed later.

4. Click **Choose File** to upload a JSON definition file.
5. Click **Save and Close** or **Save and continue**.

The Interactive Documentation item for the API definition is added in the Interactive Documentation list.



Configuring Portal Domains

The **Portal Domains** page allows you to add domain names to the Developer Portal.

Home / Portal Domains		
Portal Domains		
my search terms Search Page 1 of 1 + New		
Domain	Created	Actions
portal.mashery.com (Primary)	Nov 17 2021 02:02	

Adding a New Portal Domain

1. On the **Portal Domains** page, click **New**.
2. In the **Domain Settings** window, set the following fields:
 - **Name** - name for the portal.
 - **Primary Domain** - Select this to make this domain your primary domain.
 - **Captcha Settings** - Select the **Enable Captcha** checkbox to enable Captcha on your portal.

3. Click **Save**.

Configure SMTP/Admin Emails

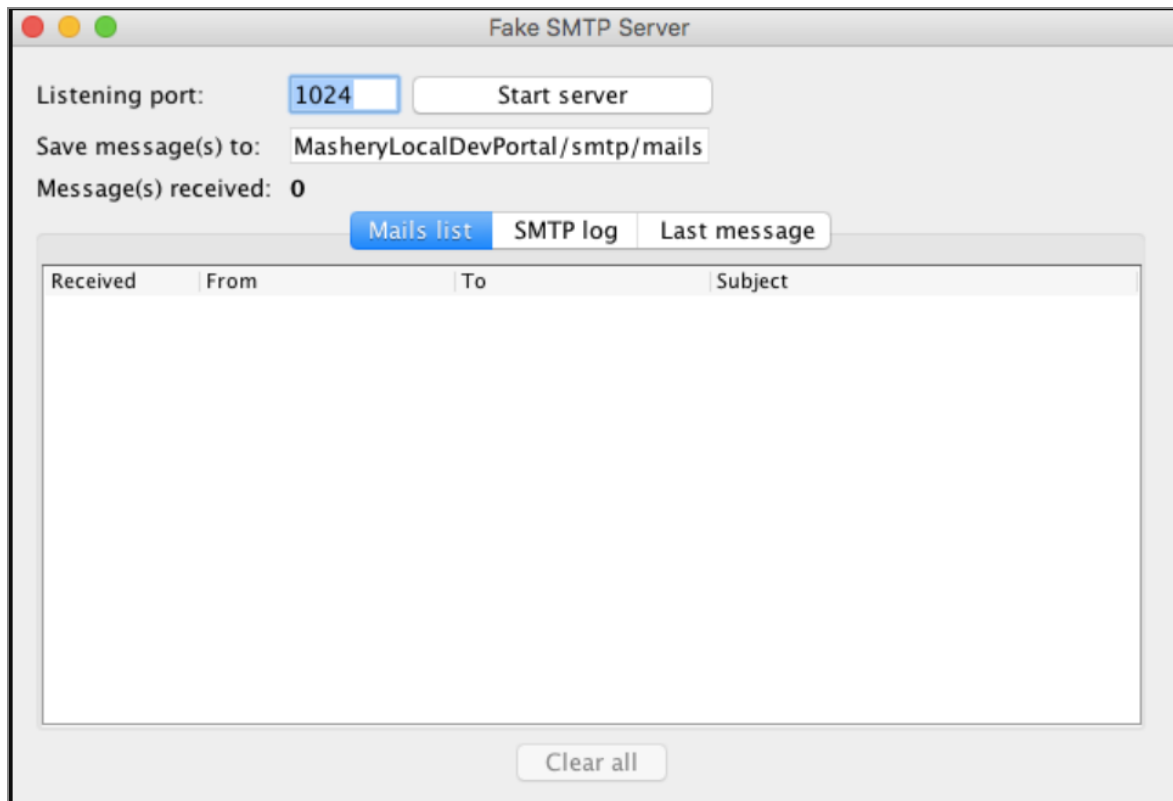
This section provides the information for the SMTP configuration with the ML setup taking the tool fake SMTP to test SMTP/admin emails for portal notifications.

Download the Fake SMTP and Start the Fake SMTP Service:

FakeSMTP is a Free Fake SMTP Server with GUI for testing emails in applications easily. It is written in Java.

Setup:

1. Download the fake SMTP jar from <http://nilhcem.com/FakeSMTP/download.html>.
2. Go to the Downloaded jar folder and start the fake SMTP service:
<http://nilhcem.com/FakeSMTP/download.html>
3. Start the fake SMTP service with the command: `java -jar fakeSMTP-2.1-SNAPSHOT.jar`
4. A service will start as follows.



SMTP setup on ML Cluster

1. Get the IP and port and other required information for the SMTP server.
2. Logon to the Boomi Cloud™ API Management - Local Edition CM container.
3. Go and open the properties file at the location on cm container:
/opt/mashery/containeragent/resources/properties/tml_cm_properties_final.json
4. Update the following property values. :

"smtp_host" : "SMTP host server ip address"

"smtp_port" : "port on the SMTP server listening"

"smtp_user" : ""

"smtp_password" : ""

"smtp_useTls" : "false"

eg :

"smtp_host" : "10.98.50.160"

"smtp_port" : 1024,


```
"smtp_user" : ""
```

```
"smtp_password" : ""
```

5. Start the jetty service on CM container using the command: `service jetty restart`
6. Test the SMTP service by making a curl call from the CM node: if the SMTP server is correctly configured then there should be mails in the SMTP server.

```
curl -X POST \
  http://localhost:4080/tml-mail/api/v1/mail/json \
  -H 'Content-Type: application/json' \
  -H 'Postman-Token: 89088130-7ddb-433e-a005-7a907e0855ff' \
  -H 'cache-control: no-cache' \
  -d '{ "sender": "ankitjaisender@tibco.com", "ccRecipients": ["cc@tibco.com"], "bccRecipients": ["bcc@tibco.com"], "recipients": ["to@tibco.com"], "subject": "test subject", "body": "testing out mails service" }'
```

Managing Portal Settings

The Portal Settings page allows you to configure and manage settings for your Developer Portal.

Use the **Footer** field to configure custom content for the footer that appears at the bottom of your Developer Portal pages.

The screenshot shows the 'Footer Content' configuration page. On the left is a sidebar with navigation links: Primary Navigation, Footer Navigation, Footer Content (selected), Email Templates, New Member Registration, New Key Issued, and New Application Created. The main content area has a breadcrumb 'Home / Footer Content' and a title 'Footer Content'. Below this is a 'Footer Content' section with a text area labeled 'Footer' and a 'Save' button. To the right of the text area is instructional text: 'Use this field to define markup that you want to emit into the footer of every developer portal page. You can include footer navigation items by adding {{footerNav}} to your template.'

Customizing Registration Email Body

Local Edition provides an option to customize the registration email body.

To customize the email body:

Procedure

1. In the **Configuration Manager** window, click **Portal Configuration**.
The **Primary Navigation** window opens.
2. Click **Email Templates** tab and then click **New Member Registration**.
3. In the **New Member Registration** window type the following:
 - From Email
 - From Name
 - Bcc Email
 - Subject
 - Body
 - Admin Notify Email

The screenshot shows the 'New Member Registration' configuration window. On the left is a sidebar with 'Primary Navigation' and 'Email Templates' sections. The 'Email Templates' section is expanded, showing 'New Member Registration' as the selected template. The main area contains fields for 'From Email', 'From Name', 'Bcc Email', 'Subject', 'Body', and 'Admin Notify Email'. Each field has a text input and a descriptive tooltip. The 'From Email' field is set to 'noreply@mashery.com'. The 'From Name' field is also 'noreply@mashery.com'. The 'Bcc Email' field is 'noreply@mashery.com'. The 'Subject' field is 'Account confirmation'. The 'Body' field contains a confirmation message with placeholders for the portal URL and a confirmation link. The 'Admin Notify Email' field is 'noreply@mashery.com'.

Field	Value	Description
From Email	noreply@mashery.com	This is the email address the notification will be sent from.
From Name	noreply@mashery.com	This is the name that will be shown for the "From" email address.
Bcc Email	noreply@mashery.com	Comma separate multiple email addresses. Example: monkey@banana.net, wombat@animals.com, giraffe@animals.com.
Subject	Account confirmation	This is the subject used in the emails that go out.
Body	A user account has been created at %URL% using this email address. If you have not registered an account, you can ignore this message. To confirm your account, please click the link below. Confirm Account %LINK%	This is the body of the Confirmation emails that get sent out to your new developers. %URL% - Portal URL %LINK% - Confirmation link to activate user
Admin Notify Email	noreply@mashery.com	Comma separate multiple email addresses. Example: monkey@banana.net, wombat@animals.com, giraffe@animals.com. This will notify admins along with developers.

Managing Organizations

One of the biggest challenges to implementing a least-privilege approach to API management assets is the need for a way to model your API assets' ownership in a way that makes sense for your business. To help you map your business into our management service, we've developed a new type of asset/entity into the Local Edition - 'Organizations'. Organizations are logical subdivisions of your overall deployment... a way for you to create an API asset management hierarchy that maps to organizations in your company.

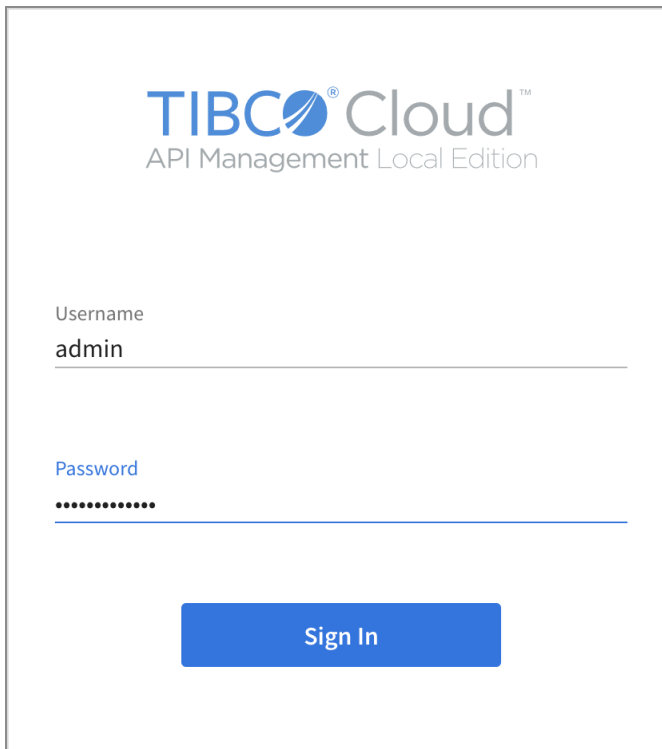
Any number of organizations can be created in your deployment, and each of those organizations can have any number of 'child' or 'sub'-organizations. Those two levels of depth should allow you to model, at a high level, the structure of those teams operating inside of your deployment -- and the roles that are generated with each organization will allow you to grant members of your deployment-specific, filtered access to only those assets that they are entitled to manage based on their Organization membership.

It's important to place API assets into the ownership of specific organizations where possible, as an important step in furthering a least-privilege model of security access -- it also reduces the likelihood of human error or unintentional changes when too many people have access to API assets that are irrelevant to their responsibilities, and reduces the amount of information people need to sift through, by filtering the server response to only those assets relevant to the user.

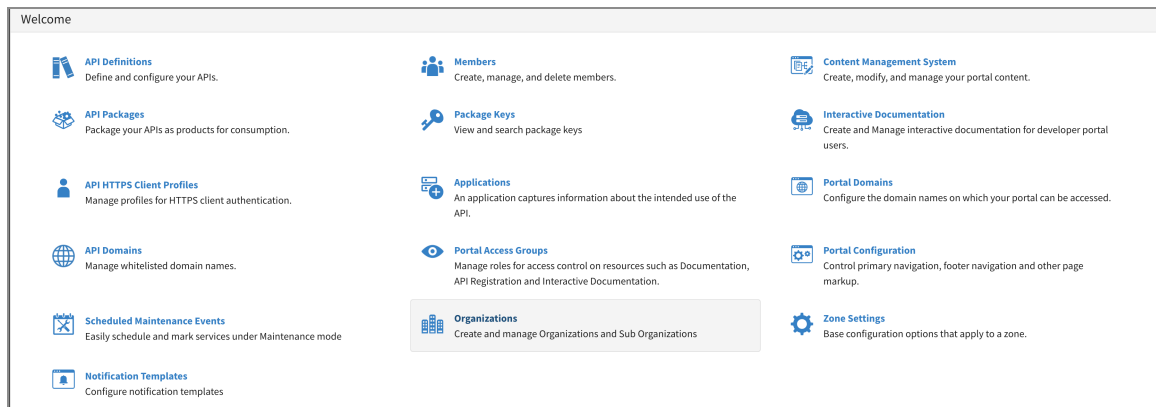
Creating Organizations

To create your first organization:

1. Log in to the Configuration Manager application.

The image shows a login interface for TIBCO Cloud API Management Local Edition. At the top, the logo "TIBCO® Cloud™" is displayed in blue, with "API Management Local Edition" in a smaller, grey font below it. Below the logo, there are two input fields. The first is labeled "Username" in a small grey font, and the text "admin" is entered. The second is labeled "Password" in a small blue font, and the password is represented by a series of dots. Below these fields is a blue rectangular button with the text "Sign In" in white.

2. Click on **Organizations** on the Configuration Manager welcome page.



3. Click .

4. Name and describe your organization, and indicate where in the hierarchy you want to place this organization by naming its parent node in the tree structure.

5. Click .

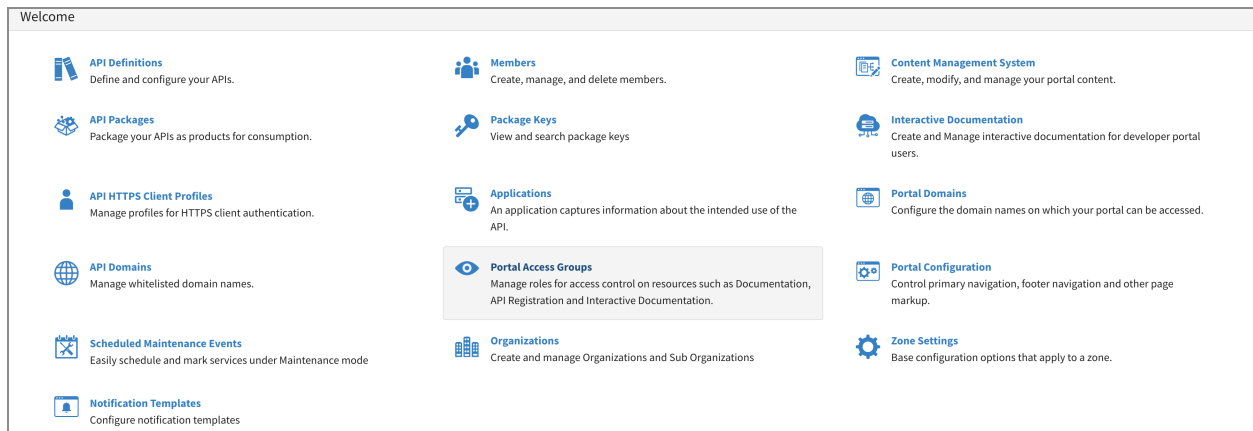
6. Create clean 'separations-of-concern' between your various API programs! Begin assigning ownership of API assets to organizations by assigning members to these organizations, creating content repositories for those organizations, and moving area-level API assets into smaller, more focused organizational groupings.

Assigning Organization Administration Roles

After defining an organization, you may notice that your area/deployment has some new roles available; these are provided to help you start filtering area members into specific

organizations for the purpose of enabling management of organization-assigned assets.

To view your currently-defined roles, click on **Portal Access Groups** on the Home page:



You will see a list of all the portal access groups you have available for role assignment:

Home / Portal Access Groups			
Portal Access Groups		my search terms Search	Page 1 of 2 + New
Name	Created	Updated	Actions
Building Services Reports User	Nov 17 2021 19:22	Nov 17 2021 19:22	
Building Services Support User	Nov 17 2021 19:22	Nov 17 2021 19:22	
Building Services API Manager	Nov 17 2021 19:22	Nov 17 2021 19:22	
Building Services Community Manager	Nov 17 2021 19:22	Nov 17 2021 19:22	
Building Services Administrator	Nov 17 2021 19:22	Nov 17 2021 19:22	
Corporate Infrastructure Reports User	Nov 17 2021 19:21	Nov 17 2021 19:21	
Corporate Infrastructure Support User	Nov 17 2021 19:21	Nov 17 2021 19:21	
Corporate Infrastructure API Manager	Nov 17 2021 19:21	Nov 17 2021 19:21	
Corporate Infrastructure Community Manager	Nov 17 2021 19:21	Nov 17 2021 19:21	

You'll notice that you cannot modify the portal access groups created by the system, only those that you have defined yourself -- don't worry, this is intentional, as these are provided as a complement to organizations and each has or will have specific implications in the context of this fast-growing application. Assigning the roles happens at the asset level -- let's look at a few of the places where these roles make a big difference:

- **API Definitions** - When you assign a portal access group to an API Definition, you're stating a policy dictating that "...the bearer of this role has access to the Interactive Documentation for this service." This is how you educate people about an API, and let them interact with it (even if you aren't necessarily letting them manage it). When an API Definition has been made available to a particular portal access group, if it has a Swagger/OpenAPI specification associated with it in the

Interactive Documentation management page. It will be available in the Local Edition Developer Portal at the `"/interactive-documentation"` route.

- **Plans** - When you assign a portal access group to a Plan, you're stating a policy dictating that "...the bearer of this role has access to this plan, pursuant to the plan's own configuration - this is how you give people the ability to register for an API Key in your developer portal. When a Plan has been made available to a particular portal access group, it will be available as a Registerable API in the Application Registration, and Add APIs to Application views of the developer portal.

Developer Portal
bgarretson
Home Interactive documentation Design Documents Test Plans

Create application

Name of your application

Internal Building Services Mobile App

You can change it later

Please describe what your application will do

This application will allow Building Services to manage their report generation and other event-driven activities from their mobile phone.

Website

http://business-services.dynacorp.com/about-us

Register OAuth 2.0 callback URL

Select APIs

Accounts Authentication API

Plan	Key rate limits
☒ None	NA
☐ SSO	2 5000

Build Services

Plan	Key rate limits
☒ None	NA
☐ Internal	2 5000

☐ I agree to the [terms of service](#).

Register Application

- **Content** - When you assign a portal access group to a [Content Page](#) in the [Content Management System](#), you're deciding who gets to navigate to and/or see that content in the developer portal - this is how you shape your content audiences and target specific segments of your developer population (internal, or otherwise) with useful information.

Sub-Topic: Document Navigation

The Document Navigation provided by default in the developer portal is effectively a site map that is established on a per-user basis per authenticated developer portal session. While an Administrator might see a tree in its entirety, if the Content Management System is sufficiently configured, a member of the 'Engineering' organization might only see a subtree of the total available content.

'Administrator'	'Engineering Organization Administrator'
Welcome to Dynacorp Quality Assurance ◦ Our Testing Approach ◦ Test Plans ▪ Mashery Local 5.3 Site Reliability Engineering ◦ SRE Secret Sauce ◦ Runbooks ▪ Authentication Service - Incident Response Engineering ◦ Who We Are ◦ Top Secret ▪ Don't Tell Anyone, Admin ◦ Designing for Success ◦ Secrets	Welcome to Dynacorp Engineering ◦ Who We Are ◦ Designing for Success ◦ Secrets

Capabilities

Following is an example of organization hierarchy.

Home / Organizations		
Organizations		+ New
Name	Created	Actions
Research and Development	Nov 17 2021 19:25	⚙ 🗑
Engineering	Nov 17 2021 19:25	⚙ 🗑
Quality Assurance	Nov 17 2021 19:25	⚙ 🗑
Corporate Infrastructure	Nov 17 2021 19:21	⚙ 🗑
Building Services	Nov 17 2021 19:22	⚙ 🗑

- **Organizational Role Inheritance** - The primary organizational role, provided at the time of organization creation, is 'Organization Administrator'. This indicates that the users possessing this role have the right to manage their team's/organization's API management assets - notably APIs/Endpoints, Packages/Plans, and Content authored in the CMS. It's also important to note that in each of the three tiers of system-defined roles (area/deployment roles, 1st-tier Organization roles and 2nd-tier/sub-organization roles), all roles propagate (are inherited) down the organization tree. In the sample organization structure above:
 - members possessing the area-level 'Administrator' role will inherently/explicitly be 'Organization Administrator' in all organizations defined in the deployment
 - members possessing the 'Research and Development' organization 'Organization Administrator' role will inherently/explicitly be 'Organization Administrator' for all its child organizations ('Engineering', 'Quality Assurance', and 'Site Reliability Engineering')
 - members possessing the 'Organization Administrator' role in any leaf node of the organization tree will have no rights to API management assets outside the scope of those assigned directly to the leaf-node organization ('Engineering', 'Accounts and Authentication' among others)

The main point of this is to help you understand that **you do not need to assign administrative roles at every level of the hierarchy** - roles implicitly propagate down the tree to their lower-order equivalents.

- **API Definitions** - [API Definitions](#) are one of the key assets that can be assigned to an organization. When you first create your API/service (manually, imported from a Swagger/OpenAPI specification in a JSON file, or imported directly from a live hosted Swagger/OpenAPI specification) you will be presented with a choice as to which organization will manage this asset:

The screenshot shows the 'API Definition' form in the Boomi Cloud interface. The breadcrumb navigation at the top reads 'Home / API Definitions / New'. Below this, there are three tabs: 'API Definition' (selected), 'Create Manually', 'Import From Swagger JSON', and 'Import From Swagger URI'. Under the 'API Definition' tab, there are 'Save' and 'Cancel' buttons. The form contains three main sections: 'Organization' with a dropdown menu showing 'Engineering'; 'Name' with a text input field containing 'Provisioning API'; and 'Description' with a text area containing 'This API allows provisioning of virtual services, instances and infrastructure through our multi-cloud'. To the right of the 'Name' and 'Description' fields, there is explanatory text: '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.' and 'Modify the description of the API. You can enter purpose, limitations, or other important information about the API.'

Simply indicate the organization you want to handle management of this API Definition - you can place this service anywhere in your area/deployment that makes logical sense for your business. After it has been placed in the custody of an organization, it ceases to be visible to other Dashboard users, unless they have the appropriate access (see [Organization Role Inheritance](#))

- **Packages** - Like API Definitions, Packages are critical for bundling the right APIs and endpoints together and distributing them to each developer community with a simple API Key. You may find yourself needing to consume another organization's services or even bundle them together with your own offering. With organizational structure in place, you can focus those cross-cutting decisions to the appropriate level by coordinating with an administrator in the organization that is their shared common 'ancestor' in the hierarchy (which may in fact be the area/deployment level itself) who can help create the cross-organization bundles for distribution to your API consumers.

When you create a Package you will similarly be presented with an 'Organization' decision to make:

The screenshot shows a web form titled 'API Package' with a breadcrumb 'Home / API Packages / New'. Below the title are 'Save' and 'Cancel' buttons. The form contains three main sections: 'Organization' with a dropdown menu currently showing 'Building Services'; 'Name' with a text input field containing 'Build Services - Internal' and a placeholder instruction 'Enter a Package name. This field allows you to modify the Package name.'; and 'Description' with a text area and a placeholder instruction 'Describe the Package.'.

Indicate the organization you wish to manage at this level of bundling for services. Remember that you can't tie APIs from different organizations unless you have area/deployment 'Administrator' access, or you possess the correct roles to make both organizations visible to you. Here again, after the package has been assigned to an organization, it is invisible to members who lack appropriate access (see [Organization Role Inheritance](#)).

- **Content Pages** - Content Pages or even entire Folders (generally 'Content Objects') can be designated as assets that belong to a specific organization. It's a common pattern to give organizations control of their own Documentation repository, and we support this during content creation and after creation in the [Organization settings](#) tab of the 'Advanced Options' configuration on each content object.

Organization

Area Level

Parent

Root

Order

Choose

Page

Choose

Role Based Access Control Matrix

The table below lists all the roles and their corresponding access to Menu options for the Configuration Manager:

	Admin	Content Manager	Org Report User	Org Support User	Org Api Manager	Org Community Manager	Org Admin
Api Definitions	✓	✗	✗	✓	✓	✗	✓
Members	✓	✗	✗	✓	✗	✓	✓
CMS	✓	✓	✗	✓	✓	✗	✓
Packages	✓	✗	✗	✓	✓	✗	✓
Package Keys	✓	✗	✗	✓	✗	✓	✓
Interactive Docs	✓	✗	✗	✓	✓	✗	✓
HTTP Clients Profile	✓	✗	✗	✗	✗	✗	✗
Applications	✓	✗	✗	✓	✗	✓	✓
Portal Domains	✓	✗	✗	✓	✓	✗	✓
Api Domains	✓	✗	✗	✓	✓	✗	✓
PAGs	✓	✓	✗	✓	✗	✓	✓
Portal Configuration	✓	✗	✗	✗	✗	✗	✓
Scheduled Maintenance Events	✓	✗	✗	✗	✓	✗	✓
Orgs	✓	✗	✗	✓	✓	✗	✓
Zone Settings	✓	✗	✗	✗	✗	✗	✓
Notification Templates	✓	✗	✗	✓	✓	✗	✓

Managing Schedule Maintenance Events

Home / Scheduled Maintenance Events			
Scheduled Maintenance Events		my search terms Search	Page 1 of 1 + New
Name	Start Time	End Time	Actions
Holiday Maintenance	Nov 29 2021 03:09	Nov 30 2021 03:09	 

Scheduled Maintenance Event	Home / Scheduled Maintenance Events / Endpoints / New						
Endpoints	Endpoint						
Select an API definition and endpoint to add to the scheduled maintenance event.							
Available API Definitions Page 1 of 1 <table border="1"> <thead> <tr> <th>Name</th> <th>Actions</th> </tr> </thead> <tbody> <tr> <td>Provisioning API</td> <td>Select</td> </tr> <tr> <td>HelloAPI</td> <td>Select</td> </tr> </tbody> </table>		Name	Actions	Provisioning API	Select	HelloAPI	Select
Name	Actions						
Provisioning API	Select						
HelloAPI	Select						

LDAP and SAML

This page describes how to set up LDAP and SAML servers with Local Edition in untethered mode. The following configurations enable LDAP and SAML for both Configuration Manager and Developer Portal.

- [Configuring LDAP and SAML with Local Edition](#)
- [Logging into Configuration Manager with LDAP and SAML](#)

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 LDAP Login	The text on the LDAP login button.
LDAP URL ldap://34.67.34.161:1234	The LDAP URL, starts with 'ldap://', 'ldaps://', or 'ldapi://'.
Bind Username uid=john,ou=People,dc=nodomain	The username to use for looking up users.
Bind Password johnldap	The password to use for looking up users.
Search Filter (&(objectclass=inetOrgPerson)(uid=%s))	Example: '(&(objectClass=organizationalPerson)(uid=%s))'
Base DN dc=nodomain	Example: 'ou=People,dc=nodomain'
☒ Start TLS	Use StartTLS with 'ldap://'.
☒ Skip Verify	Don't verify the certificate.
Save And Test LDAP	
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
2. Provide the LDAP Username and password and click the **Sign In** 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.

|



Note: LDAP and SAML login credential must be different.

FAQ and Troubleshooting

Question / Issue	Answer
User interface Save or Cancel does not respond.	Check if the server is running on the port that is used in the browser URL. The server must be running at the configured Configuration Manager hostname /admin.
How do I create multiple endpoints with the same path but different HTTP methods?	The business rule is: No two endpoints can have the same domain, path and method combination. But at creation time, the methods cannot be manipulated. So the workaround is to create an endpoint with a different path, change the methods and then revert the path for the endpoint. The workaround to create multiple endpoints with the same path but with different HTTP methods is provided below, and uses the following two example endpoints with these details: - Endpoint 1 - http://api.example.com/hello (Method: Get, Post) - Endpoint 2 - http://api.example.com/hello (Method: Put, Delete) Workaround: 1. Navigate to Home > API Definitions. 2. Create New API Definition. 3. Enter name 'sample api' and save the API Definition. 4. Navigate to Endpoints. 5. Click New. 6. Enter the following details for Endpoint 1: a. Name: EP1 b. Published EP Domain: api.example.com c. Published EP Path: /hello d. Origin EP Domain IP/Address: localhost e. Origin EP Path: /hello 7. Click Save. 8. Navigate to Home > API Definitions > sample api >

Question / Issue	Answer
	Endpoints. 9. Create second endpoint with the following details: a. Name: EP2 b. Published EP Domain: api.example.com c. Published EP Path: /hello_temp d. Origin EP Domain IP/Address: localhost e. Origin EP Path: /hello Note: Note that the Published EP Path is /hello_temp. This is temporary. 10. Save the details for Endpoint 2. 11. Navigate to Protocol and Authentication tab in the sidebar for EP2. 12. Select check boxes for Put and Delete and unselect boxes for Get and Post. 13. Save the changes. 14. Navigate back to Endpoint Settings tab for EP2. 15. Change the path from /hello_temp to /hello. 16. Save the details. This will give you two endpoints with same path but different methods.
How are out of the box HTTP Client Profiles set up?	The cluster admin must configure the identity and stores using the clustermanager CLI. Once they are added via that tool, the identities and trusts should show up in the admin UI immediately and be able to be mapped to newly-created HTTPS Client Profiles that can then be used for endpoints.
How do I pick a Profile in Configuration Manager?	
What types or certificates are	

Question / Issue	Answer
available and when are they used?	
How does the Local Edition admin set up identity and trust profiles?	The Cluster Manager commands to create Truststore and Identity are documented in the Cluster Manager CLI topic. The API designer can choose to make the endpoint secure using HTTPS. When a client (for example, a browser) makes a request for a secure endpoint, the Traffic Manager forwards the request to the internal backend endpoint. The backend can be one-way authenticated or can expect the client (Traffic Manager in this case) to perform mutual authentication. This can be done in two ways: 1. One-way authenticated backend endpoint. In this case, the backend's CA cert will be compared against the known Truststore. 2. Mutual authentication scenario performs the server validation, that point #1 covers, but also expects the client to be authenticated by the server.

Developer Portal

Deliver a secure, on-premises Developer Portal deployed on Local Edition's API management platform. You can use the out-of-the-box configurations (pre-defined Local Edition templates) to give your developers self-service portal access, or customize a Developer Portal yourself for a uniquely-branded experience.

Similar to the Boomi Cloud API Management Developer Portal, Local Edition's Developer Portal features include the ability to create interactive documentation and manage API keys.

Getting Started

Portal Roles

You will need a Local Edition Administrator role to configure and manage your Developer Portal.

Default Developer Portal Pages

Out of the box, the Developer Portal gives you several developer-facing integrations for interacting with your APIs:

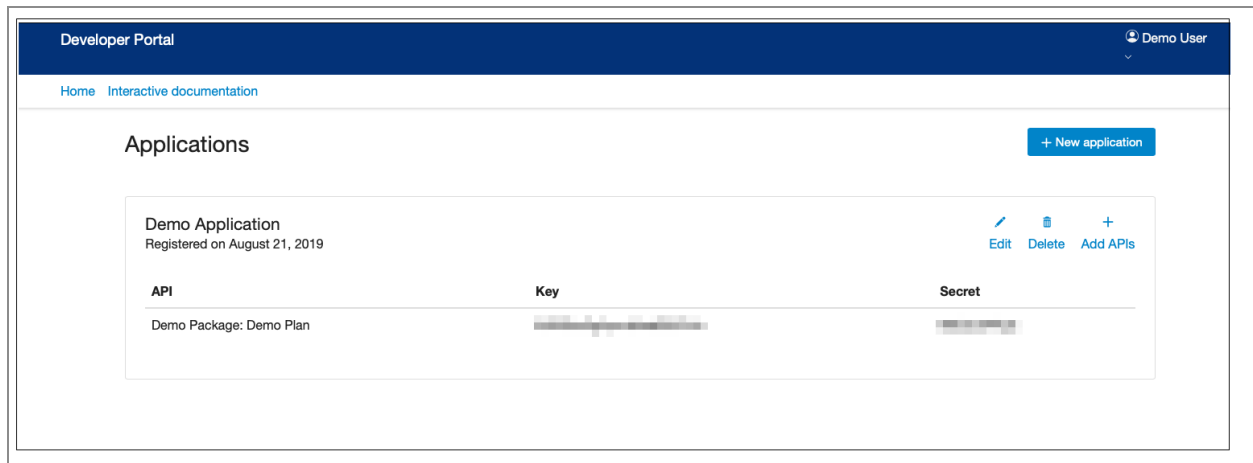
- **Home**
- **Applications**
- **Keys**
- **Interactive APIs**

Home

The **Home** page is the landing page for your Developer Portal and the gateway to your API community. Use it to welcome your users to your Developer Portal, direct them to your Developer Portal's APIs, and provide clear steps on how to become a community member, including how to register for a developer key.

Applications

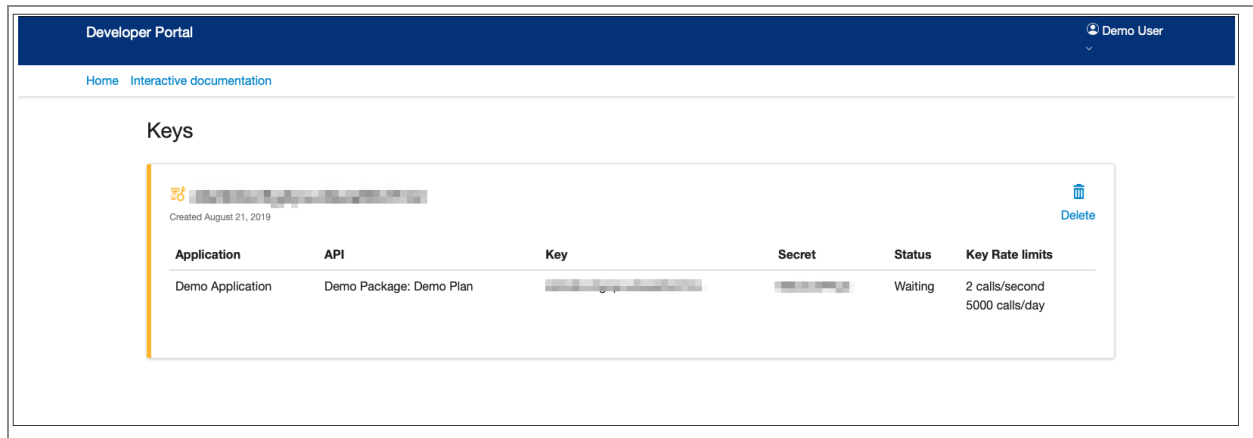
The **Applications** page displays all the registered applications.



Keys

A Package Key is given to a developer to access a set of services, endpoints, and methods as defined in the respective Package Plan. A Plan associated with a Package Key can be replaced with another Plan in the same Package without changing the key string and without requiring a developer to change the application code.

The **Keys** page displays a list of Package Keys owned by the users in the system.



Interactive APIs

The **Interactive APIs** page is an interactive documentation system for RESTful and SOAP-based APIs that are registered on the Developer Portal.

Configuring your Developer Portal Pages

Configuration for the default pages of your Developer Portal are handled through [Configuration Manager](#).

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-0-0.cache-svc-
0.default.svc.cluster.local      ACTIVE    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-0-5ccc99468f-m6s95.cm-
svc-0.default.svc.cluster.local  ACTIVE    May 24 2021 21:12:09 +0000 10.120.0.11 7080

3f9262a3-470e-4557-806e-4a23659d5fb6 logservice log-set-0-0.log-svc-
0.default.svc.cluster.local      ACTIVE    May 24 2021 21:12:19 +0000 10.120.0.12
24224
da4e3f02-3d0c-45ef-9595-25ac200f3992 nosql      cass-set-0-0.cass-svc-
0.default.svc.cluster.local      ACTIVE    May 24 2021 21:12:22 +0000 10.120.0.10 9042

afc6a10d-1ca6-4bd8-bd51-7dd413561efe sql        mysql-set-0-0.mysql-svc-
0.default.svc.cluster.local      ACTIVE    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 cache      cache-set-0-0.cache-svc-
0.default.svc.cluster.local ACTIVE    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-0-5ccc99468f-
m6s95.cm-svc-0.default.svc.cluster.local ACTIVE    May 24 2021 21:12:09 +0000
10.120.0.11 7080
```

```

3f9262a3-470e-4557-806e-4a23659d5fb6 logservice log-set-0-0.log-svc-
0.default.svc.cluster.local ACTIVE May 24 2021 21:12:19 +0000 10.120.0.12
24224
da4e3f02-3d0c-45ef-9595-25ac200f3992 nosql cass-set-0-0.cass-svc-
0.default.svc.cluster.local ACTIVE May 24 2021 21:12:22 +0000 10.120.0.10
9042
afc6a10d-1ca6-4bd8-bd51-7dd413561efe sql mysql-set-0-0.mysql-svc-
0.default.svc.cluster.local ACTIVE 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 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.

Advanced Customizations for your Developer Portal

The out-of-the-box Developer Portal UI is designed to be both functional and attractive. To customize your Developer Portal to match your brand and the unique needs of your API program, you have the following options:

- [Customize your CSS](#)
- [Modify templates and layouts](#)
- [Load custom fonts](#)
- [Configure Developer Portal settings](#)
- [Include Boomi Cloud API Management plugins](#)
- [Add third-party scripts, styles, and plugins](#)
- [Localize your Developer Portal](#)
- [JavaScript APIs](#)

Customizing your CSS

The CSS in your Developer Portal follows a consistent, predictable structure.

Loading Custom Stylesheets

You can load custom stylesheets into your Developer Portal using the `portal.loadCSS()` method.

Pass in the path of the CSS file as an argument. Add the following to your `customizations.js` file.

```
// Load a custom CSS file
portal.loadCSS('/path/to/your/file.css');
```

If you prefer, you can inline CSS directly into the document using the `portal.inlineCSS()` method. Pass in your custom CSS as a string.

```
// Custom CSS
var css =
  'p {' +
    'font-weight: bold;' +
  '};'

// Inline CSS into the document
portal.inlineCSS(css);
```

Classes and IDs

Components of the same type share one or more classes that can be used to style them all in the same way. Individual elements or components also have a unique ID that can be used to target that item specifically. For example, here is the markup for the `/signin` page.

```
<main class="main" id="main">
  <h1>Sign In</h1>
  <form id="signin-form" data-mashery="signin-form" action="">
    <label class="label label-required" id="label-username" for="username">
      <span class="label-text">Username</span>
      <input class="label-input" type="text" name="username" id="username" required>
    </label>
    <label class="label label-password label-required" id="label-password" for="password">
```

```
<span class="label-text">Password</span>
<input class="label-input" type="text" name="password" id="password" required>
</label>
<p class="form-actions">
  <button class="btn btn-primary" id="signin-form-submit">Sign In</button>
</p>
</form>
<p id="signin-recover">Forgot your credentials? <a href="/lost-password">Recover your
password</a> or <a href="/lost-username">recover your username</a>.</p>
<p id="signup-link">Don't have an account? <a href="/register">Sign up</a>.</p>
</main>
```

If you wanted to style the labels for all required elements to look the same, you could target the `.label-required` class.

If you wanted to target just the password field, you could target the `#label-password` ID.

For example:

```
/* Make all required labels red */
.label-required .label-text {
  color: red;
}

/* Make the password field label bold */
#label-password .label-text {
  font-weight: bold;
}
```

Button Classes

Buttons and action-based links (for example, to cancel an action on a page) include a common set of classes you can target for styling.

Class	Applied to...
.btn	All buttons and action-based links.
.btn-primary	The button that controls the primary action on the page.
.btn-secondary	Buttons that control secondary actions on the page.
.btn-delete	Buttons that cause apps, keys, or accounts to be deleted.

Every button also has a unique ID that you can use to override these general button

styles.

Floating Labels

Your portal uses a UI convention known as [floating labels](#) by default. You can turn this feature off by setting `floatingLabels` to `false` with the `portal.settings()` method. Add this to your `customizations.js` file.

```
// Turn off floating labels
portal.settings({floatingLabels: false});
```

Modifying Templates and Layouts

This section covers:

- [Getting Started](#)
- [Using User-Specific Data](#)
- [Creating Form Field HTML](#)
- [Adding, Removing, and Reordering Fields](#)
- [Adding Translation Strings](#)

Getting Started

You can use the `portal.setTemplate()` method to override the default template for any view in the Portal.

Pass in the ID of the view that the template is for, and the template itself. The template must be either a markup string, or a function that returns a markup string.

For example, here's the default template for the overall page layout.

```
// Overall layout
portal.setTemplate('layout', function () {
  return `
    <div id="skip-nav-wrapper" data-mashery="skip-nav"></div>
    <div id="nav-user-wrapper" data-mashery="nav-user"></div>
    <div id="nav-primary-wrapper" data-mashery="nav-primary"></div>
    <div id="notifications-wrapper" data-mashery="notifications"></div>
```

```

    <div id="main-wrapper" data-mashery="main"></div>
    <div id="footer-wrapper" data-mashery="footer"></div>`;
  });

```

If you wanted to move the user navigation *below* the primary navigation, you would do this

```

// Overall layout
portal.setTemplate('layout', function () {
  return `
    <div id="skip-nav-wrapper" data-mashery="skip-nav"></div>
    <div id="nav-primary-wrapper" data-mashery="nav-primary"></div>
    <div id="nav-user-wrapper" data-mashery="nav-user"></div>
    <div id="notifications-wrapper" data-mashery="notifications"></div>
    <div id="main-wrapper" data-mashery="main"></div>
    <div id="footer-wrapper" data-mashery="footer"></div>`;
});

```

Using User-Specific Data

Some pages, like `/applications`, require data specific to the current user to render properly.

For these views, you should use a function for the template, and provide data as an argument. The user-specific data will automatically be passed into the function when it runs.

```

portal.setTemplate('apps', function (data) {

  // Log the user-specific data to the console
  console.log(data);

  // Your template goes here...
  var html =
    '<main class="main has-apps" id="main">' +
    '<h1>Hello, world!</h1>' +
    '</main>';

  // Return the html string
  return html;

});

```

Creating Form Field HTML

If the view you're customizing includes a form, you don't have to hand-code the fields into your new layout.

The `portal.getFields()` method will return an array of field data for the form. Pass in the view ID as an argument.

For example, this would store an array of field data for the sign-in page to the field's variable:

```
// Get the field data for this view
var fields = portal.getFields('signin');
```

The `portal.getPartial('createField', field, view)` method provides a way to automatically generate markup for each of the fields.

To create the field markup, you can loop through the array of fields and pass each on into the method. Replace view with the actual view ID.

Here are examples using the JavaScript `Array.map()` method to create an array of field markup, and `Array.join()` to combine them into a single string:

```
// Get the field data for this view
var fields = portal.getFields('signin');
```

```
// Create the field markup
var fieldsHTML = fields.map(function (field) {
  return portal.getPartial('createField', field, 'signin');
}).join("");
```

Adding, Removing, and Reordering Fields

Setting fields

You can set fields form a form using the `portal.setFields()` method.

Pass in two arguments: the id of the view the fields belong with, and an array of fields. Each field in the fields array is an object with key/value pairs for the various field properties.

For example, here are the fields for the sign-in form:

```
portal.setFields('signin', [  
  {  
    id: 'username',  
    type: 'input',  
    value: null,  
    required: true  
  },  
  {  
    id: 'password',  
    type: 'password',  
    value: null,  
    required: true  
  }  
]);
```

Adding fields to an existing form

To add fields to an existing form, you first need to use the `portal.getFields()` method to get the existing fields array.

```
// Get the existing fields  
var fields = portal.getFields('signin');
```

Next, you'll use the JavaScript `Array.concat()` method to add your new field items to the default fields.

```
// Merge new fields into the existing ones  
var updatedFields = fields.concat([  
  {  
    id: 'email',  
    type: 'email',  
    value: null,  
    required: false  
  }  
]);
```

Finally, you'll use the `portal.setFields()` method to update the fields.

```
// Update with additional fields  
portal.setFields('signin', updatedFields);
```

Reordering fields

You can change the order of fields in a form using the `portal.reorderFields()` method. Pass in two arguments: the id of the view, and an array of field IDs in the order you want them to appear.

In the last section, we added an email field to the signin view. That field was added to the end of the fields array, after password.

If you wanted email to appear between username and password, you would do this:

```
// Reorder signin fields
portal.reorderFields('signin', ['username', 'email', 'password']);
```

Removing fields

You can use the `portal.removeFields()` method to remove a field from a form, as long as it's not required. Pass in the form id and an array of field IDs as arguments.

For example, if you wanted to remove two fields from the sign-in form: username and email, you would do this:

```
// Remove fields from the signin view
portal.removeFields('signin', ['username', 'email']);
```

In this case, email would be successfully removed, but username would not, since it's a required field.

Adding Translation Strings

If you have enabled multi-language support on your portal, it's important that you maintain translation strings in any of the templates that you customize. If you hard-code text into your templates, it will always show up in that language and cannot be translated.

For any text that should be translatable, pass the `viewID` and `stringID` into the `portal.translate()` method as arguments.

For example, here's the default template for the signin view:

```
portal.setTemplate('signin', function () {
  var fields = portal.getFields('signin');
```

```

return `
  <main class="main" id="main">
    <h1>${portal.translate('signin', 'title')}</h1>
    <form id="signin-form" data-mashery="signin-form" action="">
      ${fields.map(function (field) {
        return portal.getPartial('createField', field, 'signin');
      }).join("")}
      <p class="form-actions">
        <button class="btn btn-primary" id="signin-form-submit">${portal.translate('signin',
'submit')}</button>
      </p>
    </form>
    <p id="signin-recover">${portal.translate('signin', 'recover')}</p>
    <p id="signup-link">${portal.translate('signin', 'signup')}</p>
  </main>`;
});

```



Note: This example uses template literals, which do not work in older browsers. You should use string concatenation or run your templates through a tool, such as Babel, before adding to your portal.

Loading Custom Fonts

Your brand identity may include the use of a custom typeface. Your Developer Portal provides a way for you to use custom fonts.

This section covers:

- [Getting Started](#)
- [Using Fonts and Handling Performance Issues](#)

Getting Started

To use a custom typeface (a font that's not a standard system font) in your Developer Portal, you first need to load the custom stylesheet for the typeface using the `portal.loadCSS()` method.

For example, let's load the Open Sans typeface ([HTTPS://FONTS.GOOGLE.COM/SPECIMEN/OPEN+SANS](https://fonts.google.com/specimen/open+Sans)) from the Google Font Repository:

```
// Load Open Sans
portal.loadCSS('https://fonts.googleapis.com/css?family=Open+Sans');
```

Then, in your CSS, add your custom typeface as the font-family. To use the same font everywhere, you can add it to the body.

```
body {
  font-family: "Open Sans", sans-serif;
}
```

Note: When loading fonts from third-party repositories like Google Fonts, you may need to update your Developer Portal's security policies in the **Admin** area.

Using Fonts and Handling Performance Issues

In many browsers, if a custom typeface is declared but hasn't finished downloading and parsing yet, browsers will leave space for the text but not render it until the file is ready.

This is often called a Flash Of Invisible Text (or FOIT). On mobile devices and slower connections, it can leave content unusable for long periods of time ([HTTPS://WWW.IGVITA.COM/2015/04/10/FIXING-THE-BLANK-TEXT-PROBLEM/](https://www.igvita.com/2015/04/10/fixing-the-blank-text-problem/)).

To ensure that users visiting your Developer Portal have the best possible experience, you should use a system font by default, and switch to your custom font after it has been loaded.

How to Detect when a Font is Loaded

To detect when a font is loaded, use the browser-native `FontFaceSet.load()` method.

Pass in a 1em font size followed by the name of your typeface. When the font loads, append a `.fonts-loaded` class to the document:

```
if ('fonts' in document) {
  document.fonts.load('1em PT Serif').then(function () {
    document.documentElement.className += ' fonts-loaded';
  });
}
```

In your CSS, you'll load a system font by default, and switch to your custom font once

the `.fonts-loaded` class is added.

```
body {
  font-family: Helvetica, Arial, sans-serif;
}

.fonts-loaded body {
  font-family: "Open Sans", sans-serif;
}
```

How to Get Better Performance on Subsequent Page Loads

If the font repository sets far future expires headers, the typeface will be cached for subsequent page loads. Fonts hosted with Google Fonts do this.

You can improve subsequent page load performance by setting a cookie when the typeface first loads, and then automatically adding the `.fonts-loaded` class if the cookie exists.

```
var getCookie = function (name) {
  var value = "" + document.cookie;
  var parts = value.split("; " + name + "=");
  if (parts.length == 2) return parts.pop().split(";").shift();
};

if ('fonts' in document) {
  if (getCookie('fontsLoaded')) {
    document.documentElement.className += ' fonts-loaded';
  } else {
    document.fonts.load("1em PT Serif").then(function () {
      var expires = new Date(+new Date() + (7 * 24 * 60 * 60 * 1000)).toUTCString();
      document.cookie = 'fontsLoaded=true; expires=' + expires;
      document.documentElement.className += ' fonts-loaded';
    });
  }
}
```

Configuring Developer Portal Settings

Your Developer Portal comes with some out-of-the-box settings that can be changed using your `configurations.js` file.

This section covers:

- [Default Developer Portal Settings](#)
- [Changing Developer Portal Settings](#)
- [Getting Developer Portal Settings](#)
- [Updating Your Logo](#)

Default Developer Portal Settings

Here are the default Developer Portal settings.

```
var settings = {  
  
  // If true, Portal will logs errors and issues in the browser's Developer Tools console tab  
  debug: false,  
  
  // If true, forms will display floating labels  
  floatingLabels: true,  
  
  // If true, a visually hidden skip navigation link will be included for screen reader users  
  skipnav: true,  
  
  // The markup for your Portal logo  
  // Can be text, an image, or an inline SVG file  
  logo: 'Developer Portal',  
  
  // The icon set to use  
  // Options: 'default', 'alternate'  
  icons: 'default'  
  
};
```

Changing Developer Portal Settings

You can change the default Developer Portal settings using the `portal.settings()` method. Pass in an object with the key/value pairs that you want to update.

For example, to turn on debug mode and use the alternative set of icons, you would do this:

```
portal.settings({
  debug: true,
  icons: 'alternate'
});
```

Getting Developer Portal Settings

You can get an immutable copy of the current settings using the `portal.getSettings()` method:

```
var settings = portal.getSettings();
```

Updating Your Logo

Your logo can be text, an `img` element, or an inline SVG.

```
// An image element
portal.settings({
  logo: '<img alt="Acme Developer Portal" href="/path/to/our/logo.jpg">'
});

// An SVG
portal.settings({
  logo: '<svg xmlns="http://www.w3.org/2000/svg" width="1em" height="1em" viewBox="0 0 16 16"
aria-describedby="logo-title"><title id="logo-title">Acme Developer Portal</title><path
fill="#272727" d="M15.781 12.953l-4.712-4.712a.752.752 0 0 0-1.061 0l-.354.354L6.779 5.72
11.499 1h-5l-2.22 2.22-.22.22H2.998v1.061l.22.22-3.22 3.22 2.5 2.5 3.22-3.22 2.875
2.875-.354.354a.752.752 0 0 0 1.061l4.712 4.712a.752.752 0 0 0 1.061 0l1.768-1.768a.752.752
0 0 0 0-1.061z"/></svg>'
});
```

Including Boomi Cloud API Management Plugins

Your Developer Portal includes a registry of Boomi Cloud API Management-built plugins that can be activated or deactivated in your `customizations.js` file.

This section covers:

- [Getting Started](#)
- [Toggling Password Visibility](#)

Getting Started

To activate a plugin, use the `portal.plugins.on()` method, and pass in the ID of the plugin as an argument. For example, to activate the "toggle password visibility" plugin:

```
portal.plugins.on('togglePassword');
```

To deactivate a plugin, pass the plugin ID into the `portal.plugins.off()` method:

```
portal.plugins.off('togglePassword');
```

To get a list of all available plugins, use the `portal.plugins.get()` method:

```
var plugins = portal.plugins.get();
```

Toggling Password Visibility

The Toggle Password Visibility plugin adds a button to toggle the visibility of password fields to any form that contains them.

Default: on

Adding Third-party Scripts, Styles, and Plugins

You can load third-party scripts, styles, and plugins into your Developer Portal in your `customizations.js` file.

This section covers:

- [Loading Scripts](#)
- [Loading Styles](#)
- [Initializing Plugins](#)

Loading Scripts

You can load custom scripts into your Developer Portal using the `portal.loadJS()` method. Pass in the path of the JavaScript file as an argument:

```
// Load a custom CSS file
portal.loadJS('/path/to/your/file.js');
```

If a callback should run after the script loads, pass that in as the second argument:

```
// Load a custom CSS file
portal.loadJS('/path/to/your/file.js', function () {
  console.log('the file loaded!');
});
```

By default, the `portal.loadJS()` method will load JavaScript files asynchronously and does not maintain loading order. If they should be loaded in order, pass in a third argument as `true` (this can be the second argument if there is no callback function):

```
/**
 * Maintain File Load Order
 */

portal.loadJS('/path/to/your/load-first.js', function () {
  console.log('the file loaded!');
}, true);
portal.loadJS('/path/to/your/load-second.js', true);
portal.loadJS('/path/to/your/load-third.js', true);
```

Loading Styles

You can load custom stylesheets into your Developer Portal using the `portal.loadCSS()` method. Pass in the path of the CSS file as an argument:

```
// Load a custom CSS file
portal.loadCSS('/path/to/your/file.css');
```

Initializing Plugins

Many third-party plugins require initialization, or will fail if they load before the DOM (Document Object Model) is rendered. The Developer Portal emits two custom events

that you can listen for, before initializing, or loading your scripts.

Event	When it runs
prerender	After all core files and API calls have completed, but before rendering the UI.
render	After the UI has finished rendering.

For example, if you had a plugin call `createTableOfContents()` that required the DOM to be fully loaded before running, you would do this:

```
// After the UI renders
portal.on('render', '#app', function () {

  // Load the JS file
  portal.loadJS('/path/to/createTableOfContent.js', function () {

    // Once it loads, run it
    createTableOfContents();

  });

});
```

Translations

While the default language for your Developer Portal is English, it can be fully translated into any language or languages in your `translations.js` file.

This section covers:

- [Getting Started](#)
- [Updating a Single String](#)
- [Adding Multi-Language Support](#)
- [Translation Strings Reference](#)

Getting Started

Adding Your Translations

Use the `portal.addTranslations()` method to add translations for any view in the UI. It accepts three arguments: the language (`lang`), an object of translations (`translations`), and whether or not the language is right-to-left (`rtl`).

```
/**
 * Add translations for a language
 * @param {String} lang    The language ID
 * @param {Object} translations The translations object
 * @param {Boolean} rtl    If true, the language is right-to-left (optional)
 */
portal.addTranslations(lang, translations, rtl);
```

Each view that you want to translate in the UI should be a key in the `translations` object. Each view should contain an object of string IDs, and the desired text to use for them.

For example, here's the default English for the `signin` view.

```
var english = {
  signin: {

    // Page elements
    title: 'Sign In',
    submit: 'Sign In',
    submitActive: 'Signing In...',
    recover: 'Forgot your credentials? <a href="/lost-password">Recover your password</a> or <a href="/lost-username">recover your username</a>.',
    signup: 'Don\'t have an account? <a href="/register">Sign up</a>.',

    // Form fields
    username: 'Username',
    password: 'Password',

    // Message
    errorTitle: 'Sign in errors'

  }
};
```

To translate that into French, you would do this:

```
portal.addTranslations('fr', {
    signin: {

        // Page elements
        title: 'Se connecter',
        submit: 'Se connecter',
        submitActive: 'Connectez-vous...',
        recover: ' Oublié vos informations d\'identification? <a href="/lost-password">Récupère ton mot
de passe</a> ou <a href="/lost-username">récupérer votre identifiant</a>.',
        signup: 'Vous n\'avez pas de compte? <a href="/register">S\'inscrire</a>.',

        // Form fields
        username: 'Nom d\'utilisateur',
        password: 'Mot de passe',

        // Message
        errorTitle: 'Erreurs de connexion'

    }
});
```

You can find a full list of available views and string IDs in the [Translation Strings Reference](#) at the end of this section.

Updating a Single String

While translation strings are useful for translating your Developer Portal into other language, they can also be used to change words or phrases in the Developer Portal from one set of English words to another.

For example, let's say you wanted to change the phrase **Sign In** on the `signin` view to **Log In**.

The `portal.addTranslation()` method allows you to specify a translation for a single string in a specific view. Pass in the view, string id, its desired value, and the lang as arguments. Optionally, also include a boolean for `rtl` if the language is right-to-left.

```
/**
 * Add a translation for a single string
 * @param {String} view The view to add a translation to
 * @param {String} id The string ID
 * @param {String} value The string value
 * @param {String} lang The language
 * @param {Boolean} rtl If true, language is right-to-left
```

```
*/
portal.addTranslation(view, id, value, lang, rtl);
```

Here's an example.

```
// Translate a single string
portal.addTranslation('signin', 'title', 'Log In', 'en');
```

Even though your translations are still in English, you will need to run the `setLang()` method. This is because the default language ID for the Developer Portal is actually default, and you cannot write translations to that ID.

```
// Set the default language
portal.setLang('en');
```

Adding Multi-Language Support

With translations, you can add support for multiple languages and switch between them with a language picker. Here's how to do that.

1. Add your translations. For example, to add support for both French and Spanish, first add translations for both of those languages.

```
// Add translations for French
portal.addTranslations('fr', {
  signin: {
    title: 'Se connecter',
    submit: 'Se connecter',
    submitActive: 'Connectez-vous...',
    // The rest of the string IDs...
  },
  // The rest of the views...
});
```

```
// Add translations for Spanish
portal.addTranslations('es', {
  signin: {
    title: 'Registrarse',
    submit: 'Registrarse',
    submitActive: 'Iniciando sesión...',
  },
  // The rest of the views...
});
```

```
// The rest of the string IDs...
},
// The rest of the views...
});
```

2. Set your default language. If you want your default language to be any language other than English, use the `portal.setLang()` method to set it.

```
// Set the default language to French
portal.setLang('fr');
```

3. Add a language picker to your HTML template to allow users to select their own language. The most common place to do this is in the header or footer.

```
<label for="language-picker">Select Your Language</label>
<select id="language-picker">
  <option id="en">English</option>
  <option id="fr">Français</option>
  <option id="es">Español</option>
</select>
```

You can use the `portal.getLang()` method to get the current language and update the label text.

4. Detect language changes and update the UI. Use the `portal.on()` method to listen for changes to the `#language-picker` menu.

```
// Detect changes to the #language-picker menu
portal.on('change', '#language-picker', function (event) {
  // Someone changed the language selection
});
```

When that happens, use `portal.setLang()` to update the language, save the selection to `localStorage` for subsequent page loads, and use the `portal.render()` method to update the UI.

```
// Detect changes to the #language-picker menu
portal.on('change', '#language-picker', function (event) {

  // Update the language
  portal.setLang(event.target.value);
```

```
// Save their selection to localStorage
localStorage.setItem('portalLanguage', event.target.value);

// Render a new UI
portal.render();

});
```

5. Override the default language on subsequent views. Users should not have to reselect their language on each page. You can get the user's selected language from localStorage, and hook into the prerender event to update the language before the UI renders.

```
portal.on('prerender', '#app', function () {

    // Get the selected language from localStorage
    var lang = localStorage.getItem('portalLanguage');

    // If a language was chosen, use it
    if (lang) {
        portal.setLang(lang);
    }

});
```

Translation Strings Reference

This section contains all of the available views and string IDs, as well as their default English values.

```
portal.addTranslations('en', {

    /**
     * Global Strings
     */

    // The skip navigation link (for accessibility)
    skipNav: {
        link: 'Skip to main content'
    },
});
```

```

// User navigation
navUser: {
  applications: 'Applications',
  keys: 'Keys',
  manageAccount: 'Profile',
  logout: 'Log Out',
  signin: 'Sign In',
  register: 'Register'
},

// The primary navigation menu
navPrimary: {
  home: 'Home',
  interactiveDocs: 'Interactive documentation'
},

// The footer
footer: {
  copyright: `Copyright ${new Date().getFullYear()}. All rights reserved.`,
  'nav-privacy': 'Privacy Policy',
  'nav-terms': 'Terms of Use',
  'nav-contact': 'Contact Us'
},

// The inactivity popup
idle: {
  title: 'Still there?',
  message: 'For security, you will be automatically logged out shortly. Do you want to stay logged
in?',
  time: 'seconds',
  button: 'Keep me logged in'
},

```

```

/**
 * View-Specific Strings
 */

// The homepage
'_index': {
  title: 'Welcome to our developer program!',
  subtitle: 'Discover the world of our APIs.',
  btnJoin: 'Join Now',
  btnExplore: 'Learn more about our APIs',

```



```

    registerAccountTitle: 'Register',
    registerAccountDesc: 'Get an access key and usage reports for your apps.',
    registerAppsTitle: 'Build an app',
    registerAppsDesc: 'Define your application behavior and register the keys it will use.',
    callTitle: 'Make a Call',
    callDesc: 'Use our interactive documentation to make a call and see how it all works.',
    supportTitle: 'Get Support',
    supportDesc: 'Ask questions and get advice by sending an email to our support team.',
  },

  // /404
  '404': {
    title: 'Page not found',
    content: 'Sorry, the page you\'re looking for doesn\'t exist.',
    homeBtn: 'Go home'
  },

  // /500
  '500': {
    title: '500 Internal server error',
    content: 'We are unable to process your request! Please try again later.',
    homeBtn: 'Go home'
  },

  // /member/edit
  accountManage: {

    // Page elements
    title: 'Update profile',
    cancelBtn: 'Cancel',
    submitBtn: 'Update',
    changepasswordLabel: 'Password',
    changePasswordBtn: 'Change password',

    // Form fields
    username: 'Username',
    firstName: 'First name',
    lastName: 'Last name',
    displayName: 'Display Name',
    email: 'Email',
    password: 'New password',
    passwordDesc: 'Passwords must contain at least 8 characters and have at least 3 of the
following: number, uppercase, lowercase, and special character.',
    oldPassword: 'Current password',
    uri: 'Website URL',
  },

```

```

    phone: 'Phone Number',
    company: 'Company',
    address1: 'Address Line 1',
    address2: 'Address Line 2',
    locality: 'City',
    region: 'State, Province or Region',
    postalCode: 'Zip Code',
    countryCode: 'Country',

    // Success/error messages
    successMsg: 'All changes saved successfully!',
    errorTitle: 'Something went wrong'

},

// /applications/add-apis
appAddApis: {

    // Page elements
    title: 'Add APIs to {{applicationName}}',
    apisTitle: 'Select APIs',
    noAPIsContent: 'It looks like you don\'t have any registrable APIs to select from.',
    backBtn: 'Go back',

    // Form fields
    tos: 'I agree to the <a href="/terms-of-service">terms of service</a>.',
    submitBtn: 'Add APIs',

    // Table
    headingAPI: 'API',
    headingKey: 'Key',
    callsPerSecond: ' calls per second',
    callsPerDay: ' calls per day',
    selected: 'selected',
    registered: 'Registered on ',
    noAPIsTitle: 'You\'re all set',

    // Messages
    successMsg: '{{applicationName}} updated with new apis!',
    errorTitle: 'Something went wrong'

},

// /applications/create
appCreate: {

```

```

// Page elements
title: 'Create application',
submitBtn: 'Register Application',
callsPerSecond: ' calls per second',
callsPerDay: ' calls per day',
noAPIsContent: 'It looks like you don\'t have any registrable APIs to select from.',

// Form fields
apisTitle: 'Select APIs',
applicationName: 'Name of your application',
applicationNameDesc: 'You can change it later',
summary: 'Please describe what your application will do',
uri: 'Website',
oauthRedirectUri: 'Register OAuth 2.0 callback URL',
usageModel: 'How many people do you anticipate will use your application?',
selected: 'selected',
notes: 'Notes',
howDidYouHear: 'How did you hear about this API?',
tos: 'I agree to the <a href="/terms-of-service">terms of service</a>.',

// Messages
successMsg: '{{applicationName}} successfully registered!',
errorTitle: 'There was an error creating your application'

},

// /applications/delete
appDelete: {
  title: 'Delete Application',
  content: 'Are you sure you want to delete {{applicationName}}? All keys associated with this
application will be deactivated.',
  confirmBtn: 'Yes, delete this application',
  cancelBtn: 'No, keep this application',
  successMsg: "'{{applicationName}}' successfully deleted!",
  errorMsg: 'An error occurred while trying to delete {{applicationName}}.'
},

// /applications/edit
appEdit: {

  // Page elements
  title: 'Edit Application',
  cancelBtn: 'Cancel',
  submitBtn: 'Save',

```

```

// Form fields
applicationName: 'Name of your application',
summary: 'Please describe what your application will do',
uri: 'Website',
oauthRedirectUri: 'Register OAuth 2.0 callback URL',
usageModel: 'How many people do you anticipate will use your application?',
notes: 'Notes',
howDidYouHear: 'How did you hear about this API?',
tos: 'I agree to the <a href="/terms-of-service">terms of service</a>.',

// Messages
successMsg: '{{applicationName}} successfully updated!',
errorTitle: 'There was an error editing your application'

},

// /applications/summary
appSummary: {

// Page elements
title: 'API registration summary',
infoTitle: 'Application information',
applicationName: 'Name',
selectedAPIsTitle: 'Selected APIs',
backBtn: 'Back to Applications',
addAPIsBtn: 'Add APIs',
editAppBtn: 'Edit',
deleteAppBtn: 'Delete',

// Table headings
headingAPI: 'API Plan & Package',
headingKey: 'Key',
headingSecret: 'Secret',
headingStatus: 'Status',
headingLimits: 'Key rate limits',

// Table cells
uri: 'Website',
oauthRedirectUri: 'OAuth 2.0 callback URL',
appDesc: 'Description',
callsPerSecond: 'calls per second',
callsPerDay: 'calls per day',

},

```

```

// /applications
apps: {

    // Regular view
    title: 'Applications',
    createBtn: 'New application',
    addAPIsBtn: 'Add APIs',
    editBtn: 'Edit',
    deleteBtn: 'Delete',
    appName: 'Name',
    registered: 'Registered on ',

    // Tables
    headingAPI: 'API',
    headingKey: 'Key',
    headingSecret: 'Secret',

    // No Apps view
    titleNoApps: 'You don\'t have any applications yet',
    createFirstBtn: 'Register a new application',

},

// /contact
contact: {
    title: 'Contact Us',
    content: '<p>Add your contact information here.</p>'
},

// /interactive-documentation
interactiveDocs: {
    title: 'API Documentation',
    apiList: 'Select an API',
    download: 'Download',
    noDocsTitle: 'No interactive documentation available yet',
    homeBtn: 'Go home'
},

// /keys/delete
keyDelete: {

    // Page elements
    title: 'Delete key',
    content: 'Are you sure you want to delete the key {{apiKey}}? Deleting an API key that is
currently in use will most probably break the application it is used in.',

```

```

cancelBtn: 'No, keep this key',
confirmBtn: 'Yes, delete this key',

// Messages
successMsg: '{{apiKey}} deleted successfully!',
errorMsg: 'An error occurred while trying to delete {{apiKey}}. Please try again.'

},

},

// /keys
keys: {

// Standard view
title: 'Keys',
callsPerSec: 'calls/second',
callsPerDay: 'calls/day',
deleteBtn: 'Delete',

// Table headings
headingApp: 'Application',
headingAPI: 'API',
headingKey: 'Key',
headingSecret: 'Secret',
headingStatus: 'Status',
headingLimits: 'Key Rate limits',

// Empty view
noKeysTitle: 'You don\'t have any keys yet',
getKeysBtn: 'Get API keys',

},

// /lost-password
passwordRecover: {

// Page elements
title: 'Recover your password',
content: 'Enter your email or username and we\'ll send you a link to recover your password.',
submitBtn: 'Recover',
cancelBtn: 'Cancel',

// Form fields

```

```

    username: 'Username',
    email: 'Email Address',

    // Messages
    errorTitle: 'Something went wrong'

  },

  // /lost-password/submitted
  passwordRecoverSubmitted: {
    title: 'Recover your password',
    subtitle: 'Recovery request received!',
    content: 'If an account matches the username <strong>{{username}}</strong> and email
<strong>{{email}}</strong>, you should receive an email with instructions on how to reset your
password shortly.',
    typo: 'Made a typo? <a href="/lost-password">Try again</a>.',
    loginBtn: 'Go back to login'
  },

  // /password/reset
  passwordReset: {
    title: 'Reset password',
    password: 'Password',
    cancelBtn: 'Cancel',
    submitBtn: 'Submit',
    errorTitle: 'There was an error resetting your password'
  },

  // /privacy-policy
  privacyPolicy: {
    title: 'Privacy Policy',
    content: '<p>Add your privacy policy here.</p>'
  }
}

```

```

// /register
register: {

  // Page Elements
  title: 'Register for an account',
  submitBtn: 'Register',

  // Form fields
  username: 'Username',
  usernameDesc: 'After you register your username, it can\'t be changed.',
  displayName: 'Display Name',
}

```

```

        displayNameDesc: 'This is the name which other users will see',
        email: 'Email address',
        emailDesc: 'We\'ll send a validation email to this address. Validation is required to complete
registration.',
        password: 'Password',
        passwordDesc: 'Passwords must contain at least 8 characters and have at least 3 of the
following: number, uppercase, lowercase, and special character.',
        firstName: 'First name',
        lastName: 'Last name',
        company: 'Company',
        uri: 'Website URL',
        phone: 'Phone Number',
        address1: 'Address Line 1',
        address2: 'Address Line 2',
        locality: 'City',
        region: 'State, Province or Region',
        postalCode: 'Zip Code',
        countryCode: 'Country',
        tos: 'I agree to the <a href="/terms-of-service">terms of service</a>.',

        // Messages
        errorTitle: 'Registration errors'

    },

    // /register/confirmation
    registerConfirm: {
        title: 'Email verified!',
        signinBtn: 'Sign in',
    },

    // /register/submitted
    registerSubmitted: {

        // Page elements
        title: 'You\'re almost there!</h1>',
        subtitle: 'Just one more step to complete your registration.',
        content: 'Please check your <strong>{{email}}</strong> email address for a link to verify your
account.',
        resendMsg: 'Didn\'t get your confirmation email? Click the button below to resend it.',
        resendBtn: 'Resend Confirmation Email',

        // Messages
        successMsg: 'Confirmation email has been sent again successfully',
        errorMsg: 'You have reached the maximum number of tries. Your request cannot be proceeded

```



```
further'
```

```
},
```

```
// /signin
```

```
signin: {
```

```
    // Page elements
```

```
    title: 'Sign In',
```

```
    submit: 'Sign In',
```

```
    submitActive: 'Signing In...',
```

```
    recover: 'Forgot your credentials? <a href="/lost-password">Recover your password</a> or <a href="/lost-username">recover your username</a>.',
```

```
    signup: 'Don\'t have an account? <a href="/register">Sign up</a>.',
```

```
    // Form fields
```

```
    username: 'Username',
```

```
    password: 'Password',
```

```
    // Message
```

```
    errorTitle: 'Sign in errors'
```

```
},
```

```
// /signout
```

```
signout: {
```

```
    title: 'Signed Out',
```

```
    content: 'You have successfully signed out. Come back soon!'
```

```
},
```

```
// /terms
```

```
termsOfService: {
```

```
    title: 'Terms of service',
```

```
    content: '<p>Add your terms of service.</p>'
```

```
},
```

```
// /lost-username
```

```
usernameRecover: {
```

```
    // Page elements
```

```
    title: 'Recover your username',
```

```
    content: 'Enter your email and we will send you a link to recover your username.',
```

```
    submitBtn: 'Recover',
```

```
    cancelBtn: 'Cancel',
```

```

    // Form fields
    email: 'Email address',

    // Messages
    errorTitle: 'There was an issue recovering your username'

  },

  // /lost-username/submitted
  usernameRecoverSubmitted: {
    title: 'Recover your username',
    subtitle: 'Recovery request received!',
    content: 'If an account matches the email <strong>{{email}}</strong>, you will receive an email
from us that contains the username associated with the address you entered.',
    typo: 'Made a typo? <a href="/lost-username">Try again</a>.',
    loginBtn: 'Go back to login'
  }
});

```

JavaScript APIs

The JavaScript APIs are categorized in the following groups:

- [API Calls and HTTP Requests](#) - Methods for making API calls and HTTP requests.
- [Events](#) - Methods for adding and removing event listeners for various DOM events.
- [Forms](#) - Methods for setting, getting, and manipulating form fields.
- [General](#) - Methods for configuring and extending the portal.
- [Navigation](#) - Methods for setting, getting, and manipulating navigation menu items.
- [Notifications](#) - Methods for displaying notifications to portal users.
- [Plugins](#) - Methods to register, de-register, activate, and disable portal plugins.
- [Rendering](#) - Methods for rendering content into the UI.
- [Sanitize](#) - Methods for sanitizing HTML template strings before rendering them into the UI.
- [Session](#) - Methods for managing the current user session.

- [Settings](#) - Methods for configuring and managing portal settings.
- [Storage](#) - Methods for storing and accessing data locally in the browser.
- [Template](#) - Methods for setting, updating, and managing UI templates.
- [Translations](#) - Methods for internationalizing the portal.
- [Utilities](#) - Helper methods to make working with the portal a bit easier.

API Calls and HTTP Requests

Methods for making API calls and HTTP requests.

http()

Makes an HTTP request.

Syntax

```
/**
 * @param {String} url   The URL to request
 * @param {Object} options Request options
 * [optional]
 * @return {Promise}     The request promise
 */
portal.http(url, options);
```

Example

```
// A basic request
var request = portal.http('https://some-url.com');
request.then(function (response) {
  console.log(response.data); // The
  console.log(response.xhr);  // The full
  console.log(response.statusText);
}).catch(function (error) {
  console.log(error.status); // The error status
  console.log(error.statusText); // The error
  console.log(error.statusText);
});

// A request with the POST method
var request = portal.http('https://some-url.com', {
  method: 'POST'
});
```

httpWithToken()

Makes an HTTP request with the user credentials token automatically included.

Syntax

```
/**
 * @param {String} url   The URL to request
 * @param {Object} options Request options [optional]
 * @return {Promise}     The request promise
 */
portal.httpWithToken(url, options);
```

Example

```
// A request with the token included
var request = portal.httpWithToken('https://some-url.com');
```

Events

Methods for adding and removing event listeners for various DOM events.

on()

Adds an event listener.

Syntax

```
/**
 * @param {String} types   The event type or types (space
 *                          separated)
 * @param {String} selector The selector to run the event on
 * @param {Function} callback The function to run when the
 *                          event fires
 * @param {Boolean} protect If true, event cannot be removed
 */
portal.on(types, selector, callback, protect)
```

Example

```
// Listen for click events on the #app element
portal.on('click', '#app', function (event) {
  console.log('it was clicked');
});
```

on()

```
// Listen for scroll and resize events on the document
portal.on('click scroll', document, function (event) {
  console.log('An ' + event.type + ' event happened');
});

// Protect a click event so it can't be removed
portal.on('click', '#my-element', function (event) {
  console.log('You cannot remove this.');
}, true);
```

off()

Removes an event listener from an element. The arguments must exactly match the ones used to set up the event.

Syntax

```
/**
 * @param {String} types The event type or types (space separated)
 * @param {String} selector The selector to remove the event from
 * @param {Function} callback The function to remove
 */
portal.off(types, selector, callback)
```

Example

```
// Remove click events on the #app element
portal.off('click', '#app', function (event) {
  console.log('it was clicked');
});

// Remove scroll and resize events on the document
portal.off('click scroll', document, function (event) {
  console.log('An ' + event.type + ' event happened');
});

// Will not remove this event, since it was protected
portal.off('click', '#my-element', function (event) {
  console.log('You cannot remove this.');
});
```

once()

Adds an event listener, and automatically removes it after it runs once.

Syntax

```
/**
 * @param {String} types The event type or types (space separated)
 * @param {String} selector The selector to run the event on
 * @param {Function} callback The function to run when the event fires
 */
portal.once(types, selector, callback)
```

Example

```
// Listen for a click event on the #app element just once
portal.once('click', '#app', function (event) {
  console.log('it was clicked');
});
```

emit()

Emits a custom event that you can listen for with one the on() and once() methods.

Syntax

```
/**
 * @param {String} type The event type
 * @param {Node} elem The element to attach the event to
 * @param {Object} detail Any details to pass along with the event [optional]
 */
portal.emit(type, elem, detail)
```

Example

```
// Emit a custom event on the document
portal.emit('myEvent', document);

// Emit a custom event with details
portal.emit('myOtherEvent', document, {
  id: 123
});

// Listen for a custom event
portal.on('myEvent', document, function (event) {
```

emit()

```

    console.log('The event ran');
  });

  // Get the details from a custom event
  portal.on('myOtherEvent', document, function (event) {
    console.log('id:', event.detail.id);
  });

```

getEvents()

Returns an immutable list of all current event listeners.

Syntax

```

/**
 * @return {Object} Active event listeners
 */
portal.getEvents()

```

Example

```

// Get an object of current events
var events = portal.getEvents();

```

Forms

Methods for setting, getting, and manipulating form fields.

removeFields()

Removes one or more fields from a form.

Syntax

```

/**
 * @param {String} id The form ID
 * @param {String|Array} fields The field(s) to remove
 */
portal.removeFields(id, fields)

```

Example

removeFields()

```
// Remove the "comments" field from "myNewForm"
portal.removeFields('myNewForm', "comments");

// Remove the "comments" and "company" fields from
"anotherForm"
portal.removeFields('anotherForm' ['comments', 'company']);
```

getFields()

Gets all of the fields in a form.

Syntax

```
/**
 * @param {String} id The form ID
 * @return {Array} The fields
 */
portal.getFields(id)
```

Example

```
// Get the fields for 'myNewForm'
var fields = portal.getFields
('myNewForm');
```

reorderFields()

Changes the default order of fields in a form. Any form fields omitted from the array of fields are added to the end in their original order.

Syntax

```
/**
 * @param {String} id The form ID
 * @param {Array} fields The fields to reorder (ex. ['password', 'username'])
 */
portal.reorderFields(id, fields)
```

Example

```
// Show the "comments" field before "password"
// "username" will be moved to the end since it was omitted
portal.reorderFields('myNewForm', ['comments', 'password']);
```


General

Methods for configuring and extending the portal.

trueTypeOf()

Gets the accurate type for an object.

Syntax

```
/**
 * @param {*} obj The object to get the
 * type of
 * @return {String} The type
 */
portal.trueType(obj)
```

Example

```
// returns "array"
var type = _portal.trueTypeOf([]);
```

extend()

Extends the portal object with new methods and properties.

Syntax

```
/**
 * @param {*} fn The function/object/etc. to add
 */
portal.extend(name, fn)
```

Example

```
// Add the sayHi() method
portal.extend('sayHi', function (name) {
  alert('Hello, ' + name);
});

// alerts "Hello, Hal"
portal.sayHi('Hal');
```

warn()

When in debug mode, logs a warning to the console. Otherwise, does nothing.

Syntax

warn()

```
/**
 * @param {String} msg The warning to log
 */
portal.warn(msg)
```

Example

```
// If in debug mode, logs "Hey, you can't do that" to the console as a warning
portal.warn('Hey, you can\'t do that!');
```

err()

When in debug mode, throws an error in the console. Otherwise, does nothing.

Syntax

```
/**
 * @param {String} msg The error message
 */
portal.err(msg)
```

Example

```
// If in debug mode, throws the following error: "Uh oh, something went wrong!"
portal.err('Uh oh, something went wrong!');
```

Navigation

Methods for setting, getting, and manipulating navigation menu items.

addNavItems()

Adds navigation items to an existing menu. Uses the allowed and disallowed properties to restrict access to a navigation item to a specific role.

Syntax

```
/**
 * @param {String} id The navigation ID
 * @param {Array} items The navigation items
 */
portal.addNavItems(id, items)
```

addNavItems()**Example**

```
portal.addNavItems('navNew', [
  {
    id: 'about',
    url: '/about-us'
  },
  {
    id: 'topSecret',
    url: '/top-secret',
    allowed: ['123', '456'],
    disallowed: ['abc']
  }
]);
```

removeNavItems()

Removes navigation items from an existing menu.

Syntax

```
/**
 * @param {String} id The nav ID
 * @param {String|Array} items The item(s) to remove
 */
portal.removeNavItems(id, items)
```

Example

```
// Remove the "about" item
portal.removeNavItems('navNew', 'about');

// Remove the "topSecret" and "home" items
portal.removeNavItems('navNew', ['home',
'topSecret']);
```

getNavItems()

Gets the nav items for a navigation menu.

Syntax

```
/**
```

getNavItems()

```

* @param {String} id The nav ID
* @return {Array} The items
*/
portal.getNavItems(id)

```

Example

```

// Get the nav items for the "navNew" menu
var navItems = portal.getNavItems
('navNew');

```

reorderNavItems()

Gets the nav items for a navigation menu. Reorders the items in a navigation menu. Items omitted from the items array are added to the end in their original order.

Syntax

```

/**
* @param {String} id The navigation menu ID
* @param {Array} items The items to reorder (ex. ['register', 'signin'])
*/
portal.reorderNavItems(id, items)

```

Example

```

// Reorder the "navNew" menu. "home" will show up last because it was omitted from the list
portal.reorderNavItems('navNew', ['topSecret', 'about', 'docs']);

```

Notifications

Methods for displaying notifications to portal users.

notifications.success()

Creates a success notification.

Syntax

```

/**
* @param {String} msg The notification message
* @param {Object} options Notification options ({title: "The Title", inline: "#inline-target", deferred:

```

notifications.success()

```
false}) [optional]
* @return {Node}      The injected notification element
*/
portal.notifications.success(msg, options)
```

Example

```
// Display a notification that says "Hello"
portal.notifications.success('Hello');

// Display a notification on the next page load
portal.notifications.success('Hello', {
  deferred: true
});

// Display a notification with a title
portal.notifications.success('Hello', {
  title: 'Congrats!'
});

// Display a notification inline in the page instead of at the top of the document
portal.notifications.success('Hello', {
  inline: '#some-form'
});
```

notifications.warning()

Creates a warning notification.

Syntax

```
/**
* @param {String} msg   The notification message
* @param {Object} options Notification options ({title: "The Title", inline: "#inline-target", deferred:
false}) [optional]
* @return {Node}      The injected notification element
*/
portal.notifications.warning(msg, options)
```

Example

```
// Display a notification that says "Hello"
```

notifications.warning()

```
portal.notifications.warning('Hello');

// Display a notification on the next page load
portal.notifications.warning('Hello', {
  deferred: true
});

// Display a notification with a title
portal.notifications.warning('Hello', {
  title: 'Congrats!'
});

// Display a notification inline in the page instead of at the top of the document
portal.notifications.warning('Hello', {
  inline: '#some-form'
});
```

notifications.error()

Creates an error notification.

Syntax

```
/**
 * @param {String} msg   The notification message
 * @param {Object} options Notification options ({title: "The Title", inline: "#inline-target", deferred:
false}) [optional]
 * @return {Node}       The injected notification element
 */
portal.notifications.error(msg, options)
```

Example

```
// Display a notification that says "Hello"
portal.notifications.error("Hello");

// Display a notification on the next page load
portal.notifications.error('Hello', {
  deferred: true
});
```

notifications.error()

```
// Display a notification with a title
portal.notifications.error('Hello', {
  title: 'Congrats!'
});

// Display a notification inline in the page instead of at the top of the document
portal.notifications.error('Hello', {
  inline: '#some-form'
});
```

notifications.info()

Creates an informational notification.

Syntax

```
/**
 * @param {String} msg   The notification message
 * @param {Object} options Notification options ({title: "The Title", inline: "#inline-target", deferred:
 false}) [optional]
 * @return {Node}       The injected notification element
 */
portal.notifications.info(msg, options)
```

Example

```
// Display a notification that says "Hello"
portal.notifications.info('Hello');

// Display a notification on the next page load
portal.notifications.info('Hello', {
  deferred: true
});

// Display a notification with a title
portal.notifications.info('Hello', {
  title: 'Congrats!'
});

// Display a notification inline in the page instead of at the top of the document
portal.notifications.info('Hello', {
  inline: '#some-form'
});
```

Plugins

Methods to register, de-register, activate, and disable portal plugins. Portal plugins are optional components that can be selectively enabled or disabled. Customers can also add their own.

plugins.add()

Registers a new plugin.

Syntax

```
/**
 * @param {String} name The plugin name
 * @param {Object} options The plugin
options
 */
portal.plugins.add(name, options)
```

Example

```
// Register a plugin
portal.plugins.add('togglePassword', {
  setup: function () {
    // Load the plugin
    portal.loadJS('/assets/js/password-
visibility.js');
  }
});

// Register a plugin but disable it by default
portal.plugins.add('togglePassword', {
  on: false,
  setup: function () {
    // Load the plugin
    portal.loadJS('/assets/js/password-
visibility.js');
  }
});
```

plugins.remove()

Removes a registered plugin.

Syntax

plugins.remove()

```
/**
 * @param {String} name The name of the plugin to
 remove
 */
portal.plugins.remove(name)
```

Example

```
// Remove the "togglePassword" plugin
portals.plugins.remove('togglePassword');
```

plugins.on()

Activates a deactivated plugin.

Syntax

```
/**
 * @param {String} name The name of the plugin to
 activate
 */
portal.plugins.on(name)
```

Example

```
// Activate the "togglePassword" plugin
portals.plugins.on('togglePassword');
```

plugins.off()

Deactivates an activated plugin.

Syntax

```
/**
 * @param {String} name The name of the plugin to
 deactivate
 */
portal.plugins.off(name)
```

plugins.off()**Example**

```
// Deactivate the "togglePassword" plugin
portals.plugins.off('togglePassword');
```

plugins.setup()

Runs a plugin's setup function. This happens automatically after the page renders, but if you manually activate a plugin after that happens, you can use this method to run it.

Syntax

```
/**
 * @param {String} name The name of the plugin to setup
 */
portal.plugins.setup(name)
```

Example

```
// Activate the "togglePassword" plugin
portals.plugins.on('togglePassword');

// Setup the "togglePassword" plugin
portals.plugins.setup('togglePassword');
```

plugins.get()

Gets a plugin or an object of all plugins.

Syntax

```
/**
 * @param {String} name The name of a specific plugin
 [optional]
 * @return {Object|Function} The plugin or list of plugins
 */
portal.plugins.get(name)
```

Example

```
// Get the "togglePassword" plugin
portals.plugins.get('togglePassword');
```

plugins.get()

```
// Get all plugins
portals.plugins.get();
```

Rendering

Methods for rendering content into the UI.

getData()

Gets an immutable copy of the data for the current page.

Syntax

```
/**
 * @return {Object} The data
 */
portal.getData()
```

Example

```
// Get a copy of the data for the current page
var data = portal.getData();
```

sanitizeData()

Sanitizes data from third-party sources before rendering it into the UI.

Syntax

```
/**
 * @param {*} data The data to sanitize
 * @param {Boolean} allowHTML If true, allow HTML in the data properties
 [optional]
 * @return {*} The sanitized data
 */
portal.sanitizeData(data, allowHTML)
```

Example

```
// Sanitize third-party data
// Returns "<h1>Hello worlds</h1>"
```

sanitizeData()

```
var sanitized = portal.sanitizeData({name: '<h1>Hello world</h1>'});

// Sanitize third-party data with HTML allowed
// Returns "<h1>Hello world</h1>"
var sanitized = portal.sanitizeData({name: '<h1>Hello world</h1>', true});
```

stringToHTML()

Converts a template string into HTML DOM nodes.

Syntax

```
/**
 * Convert a template string into HTML DOM nodes
 * @param {String} str The template string
 * @return {Node} The template HTML
 */
portal.stringToHTML(str)
```

Example

```
// Convert a string to HTML
// Returns <h1>Hello world</h1><p>How are you?</p>
portal.stringToHTML('<h1>Hello world</h1><p>How are
you?</p>');
```

render()

Renders the portal UI into the DOM. This is run automatically when the portal opens, but you can call it manually if you need the UI to re-render for some reason.

Syntax

```
portal.render()
```

Example

```
// Render/re-render the UI
portal.render();
```

Sanitize

Methods for sanitizing HTML template strings before rendering them into the UI.

sanitize()

Sanitizes an HTML string by removing malicious and unallowed properties, attributes, and elements.

Syntax

```
/**
 * @param {String} html The raw HTML string
 * @return {String}   The sanitized HTML string
 */
portal.sanitize(html)
```

Example

```
// Sanitize a string
// Returns ''
var sanitized = portal.sanitize('');
```

sanitizeNoHTML()

Removes all HTML from a template string.

Syntax

```
/**
 * @param {String} str The raw string
 * @return {String}   The sanitized string
 */
portal.sanitizeNoHTML(str)
```

Example

```
// Remove all HTML from a string
// Returns "Hello world"
var sanitized = portal.sanitize('<h1>Hello
world</h1>');
```

Session

Methods for managing the current user session.

logout()

Logs the current user out of the portal.

Syntax

```
portal.logout()
```

Example

```
// log the current user out
portal.logout();
```

getUser()

Gets one or more properties about the current user from their token.

Syntax

```
/**
 * @param {String} prop Get a specific property for the user [optional]
 * @return {Object|String} The user object or value
 */
portal.getUser(prop)
```

Example

```
// Get all user details
portal.getUser();

// Get the current user's username
portal.getUser('handle');
```

getUserRoles()

Gets the roles associated with the current user.

Syntax

```
/**
 * @return {Array} The user's role IDs
 */
portal.getUserRoles()
```

Example

getUserRoles()

```
// get the user's roles
var roles = portal.getUserRoles();
```

userHasRole()

Checks if the current user is the member of a specific role.

Syntax

```
/**
 * @param {String} id The role ID to check
 * @return {Boolean} If true, the user has the role ID
 */
portal.userHasRole(id)
```

Example

```
// Check if the user has the role with an ID of 1234
var hasAdmin = portal.userHasRole(1234);
```

getArea()

Gets one or more area details.

Syntax

```
/**
 * @param {String} prop Get a specific property for the area
 [optional]
 * @return {Object|String} The area object or value
 */
portal.getArea(prop)
```

Example

```
// get all area details
var area = portal.getArea();
```

getTimeToSessionEnd()

Checks how much time is left (in milliseconds) before the current session ends.

Syntax

getTimeToSessionEnd()

```
/**
 * @return {Integer} The number of milliseconds left in the current session
 */

portal.getTimeToSessionEnd()
```

Example

```
// Get time to session end
var ends = portal.getTimeToSessionEnd();
```

isSignedIn()

Checks if the current user is signed in.

Syntax

```
/**
 * @return {Boolean} Returns true if the user is
 signed in
 */
portal.isSignedIn()
```

Example

```
// Check if the current user is signed in
var signedin = portal.isSignedIn();
```

getUsername()

Gets the current user's username.

Syntax

```
/**
 * @return {String} The username
 */
portal.getUsername()
```

Example

```
// Get the username
```


getUsername()

```
var username =  
portal.getUsername();
```

getUserDisplayName()

Gets the current user's display name.

Syntax

```
/**  
 * @return {String} The display name  
 */  
portal.getUserDisplayName()
```

Example

```
// Get the display name  
var displayName =  
portal.getUserDisplayName();
```

getUserEmail()

Gets the current user's email address.

Syntax

```
/**  
 * Get the current user's email  
 * @return {String} The email  
 */  
portal.getUserEmail()
```

Example

```
// Get the email address  
var email = portal.getUserEmail();
```

Settings

Methods for configuring and managing portal settings.

getSettings()

Gets an immutable copy of the current portal settings, or a specific setting's value.

Syntax

```
/**
 * @param {String} setting The key of a specific setting to get [optional]
 * @return {*}           The setting or settings
 */
portal.getSettings(setting)
```

Example

```
// Get all settings
var settings = portal.getSettings();

// Get the value of the "debug" setting
var debug = portal.getSettings('debug');
```

settings()

Updates one or more portal settings.

Syntax

```
/**
 * @param {Object} options User options to override
 * settings with
 */
portal.settings(options)
```

Example

```
// Set "debug" to true and disable floating labels
portal.settings({
  debug: true,
  floatingLabels: false
});
```

Storage

Methods for storing and accessing data locally in the browser.

setStorage()

Saves data to localStorage or sessionStorage. Automatically stringifies objects and arrays.

Syntax

```
/**
 * @param {String} id    The localStorage ID
 * @param {*}      data  The data to save
 * @param {Boolean} session If true, use sessionStorage instead of localStorage
 */
portal.setStorage(id, data, session)
```

Example

```
// Saves to localStorage
portal.setStorage('lunch', {
  sandwich: 'Turkey',
  drink: 'Water',
  orderNumber: 123
});

// Saves to sessionStorage
portal.setStorage('dinner', {
  sandwich: 'Chicken',
  drink: 'Soda',
  orderNumber: 456
}, true);
```

getStorage()

Gets data from localStorage or sessionStorage. Automatically converts stringified arrays and objects back into their original format.

Syntax

```
/**
 * @param {String} id    The localStorage ID
 * @param {*}      fallback A fallback value to return if no data is found [optional]
 * @param {Boolean} session If true, get data from sessionStorage instead of localStorage
 * @return {*}          The saved data
 */
portal.getData(id, fallback, session)
```

Example

getStorage()

```
// Get data from localStorage
var stored1 = portal.getData('lunch');

// Get data from localStorage with an empty object as a fallback
var stored2 = portal.getData('lunch', {});

// Get data from sessionStorage (with no fallback)
var stored3 = portal.getData('dinner', null, true);

// Get data from sessionStorage with an empty object as a fallback
var stored3 = portal.getData('dinner', {}, true);
```

removeStorage()

Removes data from localStorage or sessionStorage.

Syntax

```
/**
 * @param {String} id    The localStorage ID
 * @param {Boolean} session If true, remove data from sessionStorage instead of
 localStorage
 */
portal.removeStorage(id, session)
```

Example

```
// Remove data from localStorage
portal.removeStorage('lunch');

// Remove data from sessionStorage
portal.removeStorage('dinner', true);
```

getCookie()

Gets the value of a cookie.

Syntax

```
/**
 * @param {String} id The cookie
 ID
```

getCookie()

```
*/
portal.getCookie(id)
```

Example

```
// Get the value of a cookie named
"date"
var date = portal.getCookie('date');
```

Templates

Methods for setting, updating, and managing UI templates.

setTemplate()

Replaces a default template for a view. If there's an endpoint associated with the view, the API data is automatically passed into the template function as an argument. The template can be a simple string, or a function that returns a string.

Syntax

```
/**
 * @param {String} id The template ID
 * @param {Function|String} template The template
 */
portal.setTemplate(id, template)
```

Example

```
// Set the template for the "newPage" page
portal.setTemplate('newPage', function (data) {
  return `
    <h1>${data.title}</h1>
    ${data.pubdate}
    ${data.content}`;
});
```

Translations

Methods for internationalizing the portal.

addTranslation()

Adds a translation for a specific view and string in the UI.

Syntax

```
/**
 * @param {String} view The view for the translation string
 * @param {String} id The translation string ID
 * @param {String} value The value for the string
 * @param {String} lang The translation language
 * @param {Boolean} rtl If true, language is RTL [optional]
 */
portal.addTranslation(view, id, value, lang, rtl)
```

Example

```
// Add a Spanish version of the "title" string on the "newPage" view
portal.addTranslation('newPage', 'title', 'Hola Mundo', 'sp');

// Add an Arabic version of the "title" string on the "newPage" view with RTL
support
portal.addTranslation('newPage', 'title', 'مرحبا بالعالم', 'sp', true);
```

addTranslations()

Adds multiple translations for one or more views in the UI.

Syntax

```
/**
 * @param {String} lang The translation language (use `null` for the default
 language)
 * @param {Object} translations The translations
 * @param {Boolean} rtl If true, language is RTL
 */
portal.addTranslations(lang, translations, rtl)
```

Example

```
// Translations
var translations = ;

// Add the translations
portal.addTranslations('sp', {
```

addTranslations()

```
newPage: {
  title: 'Hola Mundo',
  greeting: '¡Bienvenido!'
},
docs: {
  title: 'Documentación',
  greeting: 'Aprenda cómo funciona nuestro programa'
}
});
```

setLang()

Sets the current language for the portal. If the portal UI is rendered, you may want to run `portal.render()` to render a fresh version of the UI in the new language.

Syntax

```
/**
 * @param {String} lang The language (use `null` to revert to default)
 */
portal.setLang(lang)
```

Example

```
// Set the language to Spanish
portal.setLang('sp');
```

getLang()

Gets the current language for the portal.

Syntax

```
/**
 * @return {String} The language
 */
portal.getLang()
```

Example

```
// Get the current language
var lang = portal.getLang();
```

getTextDirection()

Checks if the current portal language is RTL or LTR.

Syntax

```
/**
 * @return {String} The text direction for the current
 language
 */
portal.getTextDirection()
```

Example

```
// Get the current text direction
var direction = portal.getTextDirection();
```

translate()

Gets the text in the current language for a specific view and string.

Syntax

```
/**
 * @param {String} view    The view for the translation string
 * @param {String} id      The translation string ID
 * @return {String}        The translated value
 */
portal.translate(view, id);
```

Example

```
// Get the "title" for the "newPage" view
var title = portal.translate('newPage', 'title');
```

Utilities

Helper methods to make working with the portal a bit easier.

getCountries()

Gets an array of supported country IDs and names.

Syntax

getCountries()

```
/**
 * @return {Array} Country IDs and names
 */
portal.getCountries()
```

Example

```
// Get countries
var countries = portal.getCountries();

// Returns "US"
countries[0].id;

// Returns "United States"
countries[0].name;
```

getPageID()

Gets the ID for the current view.

Syntax

```
/**
 * @return {String} The ID
 */
portal.getPageID();
```

Example

```
// Get the current page ID
var id = portal.getPageID();
```

formatDate()

Formats a timestamp into a date string.

Syntax

```
/**
 * @param {Integer} dateStr Date string
 * @return {String} A formatted date
 string
```

formatDate()

```
*/
portal.formatDate(dateStr)
```

Example

```
// Get the date from a timestamp
// Returns "January 1, 2020"
var date = portal.formatDate
(1577854800000);
```

serialize()

Serializes all of the data in a form into an object.

Syntax

```
/**
 * @param {Node} form The form to serialize
 * @param {Boolean} rawHTML If true, do not strip HTML from the
 values
 * @return {Object} The serialized form data
 */
portal.serialize(form, rawHTML)
```

Example

```
// Get a form
var form = document.querySelector('form');

// Get all of the data in a form
var formData = portal.serialize(form);

// Get all of the form data with HTML allowed
var formHTML = portal.serialize(form, true);
```

buildQuery()

Builds a query string from an object of key/value pairs.

Syntax

```
/**
```

buildQuery()

```

* @param {Object} data The data to turn into a query string
* @return {String}    The query string
*/
portal.buildQuery(data)

```

Example

```

// An object of key value pairs
var data = {
  name: 'Dee Developer',
  favoritePet: 'Dog'
};

// Build a query string
// Returns "name=Dee%20Developer&favoritePet=Dog"
var query = portal.buildQuery(data);

```

getParams()

Gets the query string parameters from a URL as an object of key/value pairs.

Syntax

```

/**
* @param {String} url The URL [optional, defaults to window.location]
* @return {Object}    The URL parameters
*/
portal.getParams(url)

```

Example

```

// Get the query string parameters from a URL
var url = 'https://myportal.com?name=Dee%20Developer&favoritePet=Dog';
var params = portal.getParams(url);

// Returns "Dee Developer"
params.name;

// Returns "Dog"
params.favoritePet;

```

loadCSS()

Loads an external CSS file into the DOM.

Syntax

```
/**
 * @param {String} url The file to load
 */
portal.loadCSS(url)
```

Example

```
// Load a CSS file
portal.loadCSS('/path/to/my/styles.css');
```

inlineCSS()

Adds inline CSS to the head element.

Syntax

```
/**
 * @param {String} css The CSS to
 * inline
 */
portal.loadCSS(css)
```

Example

```
// Inject inline CSS into the DOM
portal.inlineCSS('.font-big { font-size:
4em; }');
```

loadJS()

Loads an external JavaScript file into the DOM.

Syntax

```
/**
 * @param {String} src The file to load
 * @param {Function} cb A callback to run after the file loads
 * [optional]
 * @param {Boolean} ordered If true, do NOT load async [optional]
 */
portal.loadJS(src, cb, ordered)
```

loadJS()

Example

```
// Load a JS file
portal.loadJS('/path/to/my/scripts.js');

// Load a JS file and run a callback after it loads
portal.loadJS('/path/to/my/scripts.js', function () {
  myScripts.init();
});

// Load multiple files and preserve the source order
portal.loadJS('/path/to/my/script1.js', true);
portal.loadJS('/path/to/my/script2.js', true);
portal.loadJS('/path/to/my/script3.js', true);
portal.loadJS('/path/to/my/script4.js', true);

// Load multiple files and preserve the source order
// Run a callback after the last on eloads
portal.loadJS('/path/to/my/script1.js', true);
portal.loadJS('/path/to/my/script2.js', true);
portal.loadJS('/path/to/my/script3.js', true);
portal.loadJS('/path/to/my/script4.js', function () {
  myScripts.init();
}, true);
```

loadHead()

Add an element into the head element.

Syntax

```
/**
 * Inject HTML elements into the <head>
 * @param {String} type The HTML element type
 * @param {Object} atts The attributes and values for the element
 */
portal.loadHead(type, atts)
```

Example

```
// Add a viewport element
// <meta name="viewport" content="width=device-width, initial-
scale=1">
```

loadHead()

```
portal.loadHead('meta', [
  name: 'viewport',
  content: 'width=device-width, initial-scale=1'
]);
```

redirect()

Redirects the user to another page.

Syntax

```
/**
 * Redirect to a new URL
 * @param {String} url The URL to
 * redirect to
 */
portal.redirect(url)
```

Example

```
// Redirect to the about page
portal.redirect('/about');
```

html()

Sanitizes and renders an HTML string into the DOM.

Syntax

```
/**
 * @param {String|Node} selector The selector or node to render into
 * @param {String} html The HTML to render
 * @param {Boolean} append If true, append to DOM instead of wiping
 * existing out
 * @return {Array} An array of injected elements
 */
portal.html(selector, html, append)
```

Example

```
// Render <h1>Hello world</h1> into the #app element
portal.html('#app', '<h1>Hello world</h1>');
```

html()

```
// Add <p>How are you today?</p> in the #app element after any existing  
content  
portal.html('#app', '<p>How are you today?</p>', true);
```

Deploying Default Portal Customizations and Assets to Your Clusters

Customizing your default Developer Portal experience is one of the most important activities in your portal development process, and the specific approaches are described in [Advanced Customizations for your Developer Portal](#). Once you've developed some [translations](#), [template customizations](#) or you need to provide file assets for consumption by your markup, you have a mechanism inside your cluster by which to import those customizations into your test cluster.

This section covers:

- [Phase: Design and Development](#)
- [Phase: Production](#)

Phase: Design and Development

During design and development of your Developer Portal, you need the ability to quickly and easily see your work render in your test cluster. To help you with testing, `stage-portal-customizations` is a new command for the [Cluster Manager CLI](#) utility tool.

`stage-portal-customizations` CLI command

This CLI command can be invoked with a single parameter - a valid path to a .zip file somewhere in your tml-cm container that adheres to the [.zip file 'contract'](#), which asks for the following:

- a single .zip file contains any or all of the following types of file in the root of the .zip files' folder structure:
 - (1) file named 'customizations.js' - used for [template customizations](#).

- (1) file named 'translations.js' - used for [translation customizations](#)
- (1) folder named 'assets' - contains any arbitrary hierarchy of folders with files in them; the files can be served into your portal templates/pages by using the Files API.

Creating a valid .zip file

1. Assemble your assets, customizations.js and translations.js into a single folder on the machine where the mods were made.

For example:

```
yournamehere@My-MacBook-Pro:~/code/new-customizations|
⇒ ls -al
total 48
drwx----- 5 yournamehere staff 160 Sep 17 17:49 .
drwx-----+ 694 yournamehere staff 22208 Sep 17 16:43 ..
drwxr-xr-x@ 5 yournamehere staff 160 Sep 17 17:29 assets
-rw-r--r--@ 1 yournamehere staff 5457 Aug 6 11:53 customizations.js
-rw-r--r--@ 1 yournamehere staff 15951 Aug 6 11:53 translations.js
```

2. Create a .zip file of that folder.

For example:

```
yournamehere@My-MacBook-Pro:~/code/new-customizations|
⇒ zip -rq ~/Documents/newest-customizations.zip ./*
```

3. Copy the .zip file to your tml-cm container.

For example:

```
yournamehere@My-MacBook-Pro:~/code/new-customizations|
⇒ docker cp ~/Documents/newest-customizations.zip
a9aeb2b3f81a:/tmp/customizations.zip
```

4. Launch a Bash shell to that container from your local machine.

For example:

```
yournamehere@My-MacBook-Pro:~/code/new-customizations|
⇒ docker exec -it a9aeb2b3f81a /bin/bash
```


5. Executing the Cluster Manager stage-portal-customizations command.

For example:

```
[root@a9aeb2b3f81a mashery]# clustermanager stage-portal-customizations
/tmp/customizations.zip
Unzipped:
/opt/mashery/portal/assets
/opt/mashery/portal/customizations.js
/opt/mashery/portal/translations.js
/opt/mashery/portal/assets/images
/opt/mashery/portal/assets/images/logo.svg
/opt/mashery/portal/assets/custom.css
```

The Cluster Manager CLI will prevent you from submitting a .zip file with contents that do not adhere to these constraints, with an actionable error message to guide the remedy.

Zip File Contract Details

You can port any content or files that you want to expose in your Developer Portal, which could typically include js, css and image files. As a Local Edition Administrator, you can transport local customization files into all relevant (tml-cm) containers across any deployment, regardless of topology. The Local Edition Administrator is able to reference these ingested files in Developer Portal markup.

The .zip file must contain at its root:

- customizations.js
- translations.js (case insensitive)

Any other assets must be under a directory named assets (also case insensitive). The destination directory (/opt/mashery/portal) will be cleared before decompression is attempted and upon failure to pass validations.



Note:

customizations.js MUST be at the root (if present).

translations.js MUST be at the root (if present).

Anything else MUST be under a single folder, 'assets.' It can contain anything (js, css, mpegs, jpegs, pngs, txt, html), but it has to be under this folder (subfolders under there are also allowed).

Phase: Production

When you are satisfied with your customizations and you are preparing to deploy a new version of your Developer Portal, you will need to configure the [tml-installer build job](#), which offers a file picker with which to upload the interface-compliant ZIP file that you've been testing to-date in your test cluster. When you deploy the Production clusters, the contents of the ZIP file will be extracted into every tml-cm container you are managing during installation and all of your customizations will be immediately available.

Managing Logs in a Cluster

You can manage logs from different containers, archive dated logs, and clean up disk. Logs can also be sent to other destinations.

The following sections provide information on managing logs.

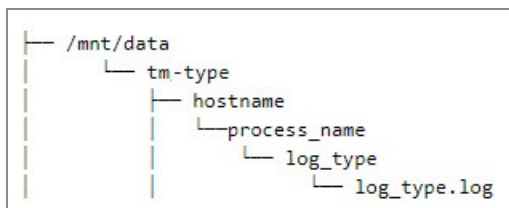
- [Viewing Logs](#)
- [Interacting with Loggers](#)
- [Collecting Logs](#)
- [Performing Log Cleanup](#)
- [Enriched Access Logs](#)
- [Exporting Logs to Different Output Channels](#)
- [Integrating with TIBCO Spotfire](#)
- [Changing Output Destinations for Different Types of Logs using CLI](#)
- [Local Edition Access Log Sync](#)

Viewing Logs

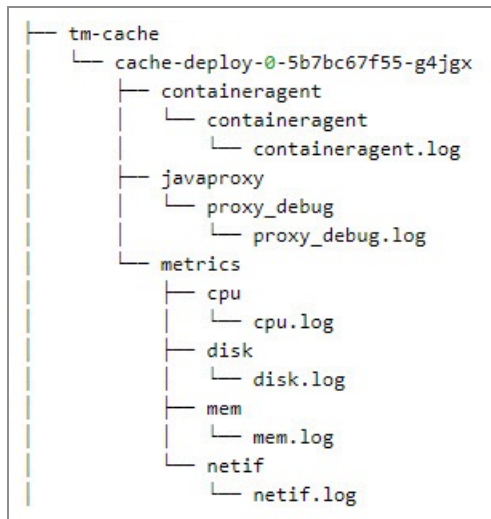
The log service stores all incoming log data (application logs, access logs and metrics) in the `/mnt/data` directory in the `tm-log` container. The data is stored in specific sub-directories.

Structure of data directory

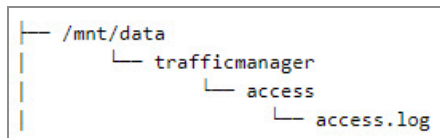
The typical directory structure for each of the Local Edition Components' log data is:



For example, the logs for the Cache service is organized as:



The exception for log data are the access logs generated by the TM-TM service. The access logs are not organized alongside the Traffic Manager container data, but kept directly under /mnt/data. Also, the access logs from different tm-tm containers would be stored at single location (there is no separation based on host name for access logs).



Interacting with Loggers

This section details using Cluster Manager CLI to interact with the different loggers in the system. The administrator can use Cluster Manager to adjust logging levels to collect more or fewer details. This section also assumes that proper clusters and zones have been set.

Most of the commands require a specific component ID. To identify the component ID, refer to the following **Example component id listing**:

Example component id listing						
clustermanager ls components						
Component ID	Component Type	Component Name	Component Status	Component Host	Component Agent Port	Component Service Port(s)
bb32be63-2a72-455e-9fffd-3e65174d119a	logservice	log	ACTIVE	10.0.0.9	9080	24224
d9927ae6-8135-4360-8031-4493e2cfd9e6	sql	sql	ACTIVE	10.0.0.11	9080	3306
d2b75938-bcd5-40e9-b260-41abb0418ba7	nosql	nosql	ACTIVE	10.0.0.5	9080	9042
fca40c79-36eb-4599-a0b4-bbe417d7b6bd	cache	cache	ACTIVE	10.0.0.13	9080	11212,11211,11213,11214,11215,11216
1bdbc338-946d-42df-a535-5f72b0144372	trafficmanager	tm	ACTIVE	10.0.0.16	9080	8080
388ca01b-1442-4a5e-856c-775325af8bbd	trafficmanager	tm	ACTIVE	10.0.0.15	9080	8080

Enabling Debug/Verbose Logging

Verbose logging is used to debug API processing by Traffic Manager.
verbose logging options and flags

```
clustermanager set verbose -h
```

Output of help

Set verbose log for traffic manager(s) in the scope of a cluster/zone/component

Usage:

```
clustermanager set verboselog [flags]
```

Aliases:

```
verboselog, verbose
```

Flags:

```
--clusterId string  cluster ID of the component
--componentId string ID of the component
--enable            Enable or disable verbose logging. 'enable=true' if the flag is empty (default true)
-h, --help          help for verboselog
--interval int      Time interval to enable verbose logging. Value must be between 1-30 (minutes)
(default -1)
--zoneId string     zone ID of the component
```

Steps to use verbose logging features

Verbose logging is a toggle switch. To re-enable verbose logging:

1. Turn off verbose logging if it was turned on.

```
clustermanager set loglevel --componentType trafficmanager --logLevel OFF --loggerName
verboselogging
```

2. Turn on verbose logging.

```
clustermanager set verbose --enable=true --interval 15
```

You can use the following to verify if verbose logging is turned on or off:

```
clustermanager ls loggers --componentType trafficmanager --componentId <componentid>
```

The following shows example output from the logging verification command:

```
clusterId [1cac153d-48db-4d5c-b3fc-9d25673ee536] and zoneId [a127e639-737c-4220-b42f-
da9f14e939dc]
Using cluster name [Local Edition Reference Cluster]
Using Zone name [local]
Using Component ID [1bdbc338-946d-42df-a535-5f72b0144372] of type [trafficmanager]
LoggerName          LogLevel Process
-----
com.mashery.tml.container.agent      INFO
com.mashery.tml.log                  INFO
com.mashery.tml.log.access.consumer  INFO
com.mashery.proxy                    INFO
com.mashery.cassandra.client         INFO
com.mashery.oauth2                   INFO
com.mashery.trafficmanager.sdk       INFO
com.mashery.internal.http            INFO
verboselogging                       INFO
```

In the above output, notice that "verboselogging" is set to INFO, that is, ON.

If you disable the verbose logging, the "verboselogging" value will be set to OFF.

Verbose-log-path: /mnt/data/trafficmanager/verbose

Verbose Log Path:

1. Connect to tml-log container
2. Go to /mnt/data/trafficmanager/verbose path
3. Folders with the POD_IP of tm containers would be created and it would have folder with request_time_epoch_requestId
e.g. `/mnt/data/trafficmanager/verbose/1542644771940-eb336334-9c83-4590-b451-d9b340539ca4.log`
4. The above file path would have all the four(inboundProcessed, inboundRequest, outboundProcessed, and outboundResponse)request in same log file.

Disable Verbose Logging:

```
clustermanager set verbose --enable=false --interval 0
```

Note:

- Verbose feature is used to debug or trace the complete call. It is not developed for high QPS running system. To use this feature, ensure that the overall worker count is increased and assign more workers to the containerlog worker.
- This is a disk extensive task. When a user wants to re-run it ensure to delete the verbose data from the previous run.



Warning: Verbose is a disk extensive operation. If the QPS is high, more number of files are generated and the CPU% increases at Log Pod hence it is advised to use this feature wisely. Each OS has restriction over total number of files. The default number of file is 1024*1024. If the number of files goes beyond 1048576, there may be some data missing in the verbose folder. For more information, see <https://access.redhat.com/solutions/6974793>

Changing log levels in containers

The administrator can change log levels of different loggers using Cluster Manager.

Set loglevel command help:

```
set loglevel -h
Set log level in the scope of a cluster/zone/component

Usage:
clustermanager set loglevel [flags]

Flags:
  --clusterId string    cluster ID of the component
  --componentId string  ID of the component
  --componentType string type of the component, valid types [sql, nosql, cache, trafficmanager,
logservice]
  -h, --help            help for loglevel
  --logLevel string     New log level, valid types [DEBUG, TRACE, INFO, WARN, ERROR]
  --loggerName string   Name of the logger to change log level
  --zoneId string       zone ID of the component
```

Listing loggers

Listing loggers on a specific component:

```
clustermanager ls loggers --componentType trafficmanager --componentId <component id>
```

Output of listing loggers command:

```
clusterId [1cac153d-48db-4d5c-b3fc-9d25673ee536] and zoneId [a127e639-737c-4220-b42f-
da9f14e939dc]
Using cluster name [Local Edition Reference Cluster]
Using Zone name [local]
Using Component ID [1bdbc338-946d-42df-a535-5f72b0144372] of type [trafficmanager]
LoggerName          LogLevel Process
-----
com.mashery.tml.container.agent      INFO
com.mashery.tml.log                  INFO
com.mashery.tml.log.access.consumer  INFO
com.mashery.proxy                    INFO
com.mashery.cassandra.client          INFO
com.mashery.oauth2                    INFO
com.mashery.trafficmanager.sdk        INFO
com.mashery.internal.http             INFO
verboselogging                       INFO
```

Changing log level of specific logger

Note the logger name from above list from running the list loggers command.

Changing the log level of a logger:

```
#This sets log level to INFO for com.mashery.tml.container.agent on all components of type
logservice. Optionally pass componentId flag to change log level on specific component
cm set loglevel --loggerName com.mashery.tml.container.agent --logLevel INFO --componentType
logservice
```

Verifying log level change:

```
clustermanager ls loggers --componentType logservice --componentId bb32be63-2a72-455e-9ffd-
3e65174d119a
```

Changing Log sync interval

The log sync interval is used by the sync process which sends logs to the reporting service on Boomi API Cloud™ Management. It is the periodic interval and end of which the sync process sends logs. The default value is 15 minutes.

To change the log sync interval:

```
cm set logsyncinterval --syncInterval 5 --componentType logservice --componentId bb32be63-2a72-455e-9ffd-3e65174d119a
```

Collecting Logs

On the log service containers, a script is available to compress logs collected and put at output path mentioned below. Container and access logs can be collected between two dates.

OUTPUT PATH: /mnt/data

Options	Possible Values	Default
-c (--container-type)	ALL or any container name (e.g. tml-log,tml-sql,tml-nosql,tml-tm,tml-cache,tml-cm)	ALL
-n (--container-hostname)	ALL or any hostname for the selected container-type. It won't allow to have only hostname passed without selecting container-type.	ALL
-a (--access-log)	No need to pass any value. If this param is selected then access logs would also be collected in separate file.	NA
-t (--log-type)	It can be ALL, METRIC, CONTAINER, VERBOSE	ALL
-s (--start-date)	It represents the start date of the log collection. Format is yyyy_mm_dd	NA
-e (--end-date)	It represents end date till what logs need to be collected. Format is yyyy_mm_dd	NA
-h (--help)	NA. Would print usage details for this utility	NA

For example, filenames for all the logs would be all-logs_2018_11_20_14_32.tar.gz or access-logs_2018_11_20_14_32.tar.gz or tm-log-log-set-0-0-logs_2018_11_20_14_32.tar.gz.

Scenario 1: To get all the logs

1. Connect to log container.
2. Run the following command:

```
/opt/mashery/containeragent/bin/getlogs.sh
```

3. Download the tar file from the /mnt/data folder, which will have a name similar to all-logs_2018_11_20_14_32.tar.gz.

Scenario 2: To get only one container type's logs from all hosts

1. Connect to log container.
2. Run the following command:

```
/opt/mashery/containeragent/bin/getlogs.sh --container-type tml-sql
```

3. Download the tar file from the /mnt/data folder, which will have a name similar to tml-sql-ALL-logs_2018_11_20_14_32.tar.gz.



Note: By default, access logs are not part of the above tar.

Scenario 3: To get logs from one container type's one host

1. Connect to log container.
2. Run the following command:

```
/opt/mashery/containeragent/bin/getlogs.sh --container-type tml-sql --container-hostname mysql-set-0-1
```

3. Download the tar file from the /mnt/data folder, which will have a name similar to tml-sql-mysql-set-0-1-logs_2018_11_20_14_32.tar.gz".
4. User can also collect logs between two dates by using start-date and end-date option.

```
/opt/mashery/containeragent/bin/getlogs.sh --container-type tml-sql --container-hostname
```

```
mysql-set-0-1 --start-date 2019_08_13 --end-date 2019_08_14
```

i Note: By default, access logs are not part of the above tar.

Scenario 4: To get metric logs only

1. Connect to log container.
2. Run the following command:

```
/opt/mashery/containeragent/bin/getlogs.sh --log-type METRIC
```

3. Download the tar file from the /mnt/data folder, which will have a name similar to All-logs_2019_08_14_06_56.tar.gz.
4. User can also collect logs between two dates by using start-date and end-date option.

```
/opt/mashery/containeragent/bin/getlogs.sh --log-type METRIC
```

i Note: By default, access logs are not part of the above tar and only metric logs would be added to tar file.

Scenario 5: To get access logs

1. Connect to log container.
2. Run the following command:

```
/opt/mashery/containeragent/bin/getlogs.sh --access-log
```

3. Download the tar file from the /mnt/data folder, which will have a name similar to access-logs_2018_11_20_14_32.tar.gz.
4. User can also collect logs between two dates by using start-date and end-date option.

```
/opt/mashery/containeragent/bin/getlogs.sh --container-type tml-sql --start-date 2019_08_13
--end-date 2019_08_14
```

i Note: User cannot use the access log option with any of the above scenario for getting access logs. Either access logs or other logs can be collected, not both

Scenario 6: Wrong selection

You cannot select container-type as ALL and specific hostname.

Example 1:

```
/opt/mashery/containeragent/bin/getlogs.sh --container-hostname mysql-set-0-1
ERROR: Please provide proper input. --container-hostname|-n can be given with --container-type|-c
```

Example 2:

```
/opt/mashery/containeragent/bin/getlogs.sh --container-type ALL --container-hostname mysql-set-0-1
ERROR: Please provide proper input. --container-hostname|-n can be given with --container-type|-c
```

You cannot collect both access logs and other logs together.

Example:

```
/opt/mashery/containeragent/bin/getlogs.sh -a --container-type tml-sql

ERROR: Access logs and other logs can not be collected at the same time. Only one type of logs can be collected
```

Performing Log Cleanup

Log cleanup is done by an automated process running in the log container at regular intervals. This clean up utility cleans the logs according to the configuration of disk.usage.warning and disk.usage.critical levels. By default, these values are set to

- disk.usage.warning=70

- disk.usage.critical=80

This number represents the percentage of the total disk allocated to the log container.

Log cleanup cron can be found at `/opt/mashery/containeragent/quartz/jobs/log-cleanup-jobs.xml`.

log-cleanup-cron

```
* /15 * * * * root /usr/bin/flock -w 0 /var/lock/logcleanup-task /usr/bin/python
/opt/mashery/containeragent/bin/cleanup_disk.py 2>&1 | tee -a /var/log/logcleanuu
pcron.log
```

You can change the disk warning and critical level in the file

`/opt/mashery/containeragent/bin/logservicecleanup.ini`

logservicecleanup.ini

```
[PATH_NOT_TO_BE_DELETED]
#please provide other log path if those logs needs to be excluded from deletion. But would be
deleted after 31 days or
# size of that dir is reached the limit allowed for that log path.
# e.g. /mnt/log1,/mnt/log2
log.path=

[BASE_PATH]
# base path for the log directory
# e.g. /mnt/data
base.path=/mnt/data

[OTHER_LOG]
# duration for which the logs need to be kept. It is in days
# This would keep the other logs of log container.
other.log.duration=1
other.log.path=/var/log/,/tmp

[CONTAINER_LOG]
# duration for which the logs need to be kept. It is in days
# This would keep the application logs of all the containers
container.log.duration=4

[METRIC_LOG]
# duration for which the logs need to be kept. It is in days
# This would keep the metrics log for the below mentioned days
metric.log.duration=4
```

[ACCESS_LOG]

duration for which the logs need to be kept. It is in days
 # This would keep the access logs for the below mentioned days
 access.log.duration=31
 access.log.path=/mnt/data/trafficmanager

[DISK_USAGE_MONITOR]

#This is the level set for cleaning up logs. Below values represents % of the configured disk
 disk.usage.warning=70
 disk.usage.critical=80

1. **PATH_NOT_TO_BE_DELETED** : It is the path which user doesn't want utility to clean up for later debugging purpose. This has to be given in comma separated value.

2. **BASE_PATH** : This is the base path where the logs are stored.

3. **OTHER_LOG** : This is the section for log container's /var/log/ and /tmp logs and "other.log.duration" property is used to define days for which user wants to keep the logs if the disk usage is above warning level.

4. **CONTAINER_LOG** : This is the section for application logs of containers and "container.log.duration" property is used to define days for which user wants to keep the logs if the disk usage is above warning level.

4. **METRIC_LOG** : This is the section for all the metrics logs and "metric.log.duration" property is used to define days for which user wants to keep the logs if the disk usage is above the warning level.

5. **ACCESS_LOG** : This is the section for all the access logs and "access.log.duration" property is used to define days for which user wants to keep the access logs if the disk usage is above critical level.
 "access.log.path" property is used to define the base path for the access logs storage.

6. **DISK_USAGE_MONITOR** : This is the section to define the disk usage warn and critical level. This represents the percentage of the disk usage.
 "disk.usage.warning" and "disk.usage.critical" properties to define the threshold level as per disk usage.

NOTE : Threshold level is totally dependent on inflow of logs and the volume of disk that is configured. As the log cleanup utility would run every 15 minutes log clean up would be continuous exercise. But user is free to set the above parameters as per the need and requirements.

Log clean up configurations

These configuration can be provided in the `tml_log_properties.json` file before starting the deployment or can be set using the cluster manager after the cluster is active. The

log clean up policy and configuration dictate threshold of cleanup and it's occurrence. The clean happens only on utilization thresholds, and not time or date.

The following configurations are applicable only, when deletion happens.

Key	Description	Data Type and Expected Values	Default Value	Example
disk_usage_warning	A percentage threshold of current disk utilization for deleting all logs except access logs.	Positive integer (ideally > 60%)	70	disk_usage_warning="70"
disk_usage_critical	A percentage threshold of current disk utilization for deleting all logs including access logs.	Positive integer (ideally > 70% and less than 95)	80	disk_usage_critical="80"
container_log_duration	Retention period (in days) of container logs. A value of 0, means delete everything.	Positive Integer	4 days	container_log_duration="1"
metric_log_duration	Retention period (in days) of metrics of different services from different containers. A value of 0, means delete everything.	Positive Integer	4 days	metric_log_duration="1"
access_log_duration	Retention period (in days) of access logs. A value of 0, means delete everything.	Positive Integer	31 days	access_log_duration="5"
payloads_duration	Retention period (in days) of payload logs.	Positive Integer	1 day	payloads_duration="2"

Key	Description	Data Type and Expected Values	Default Value	Example
	A value of 0, means delete everything.			

Enriched Access Logs

Log service generates the enriched access logs in the below format separated by ",". Currently only CSV format is supported for output enriched logs.

Field	Type	Description	Example
api_key	String	api_key used in the call	bt6pnhuqhwew22xy7jmw7bar
api_method_name	String	API method called	GetCompanyDetailRequest 34
bytes	number	Bytes we returned in the response	6668
cache_hit	number	Boolean value for whether or not we responded from the cache. 1=we did, 0=we didn't	0
client_transfer_time	number		0.00200
connect_time	number	Of the remote_total_time, how long, in seconds, it took to establish a connection.	0.001099
endpoint_	String	Endpoint name at Local	Endpoint API 2

Field	Type	Description	Example
name		Edition for which the call is made	
http_method	String	HTTP request method (GET, POST, PUT, DELETE, HEAD)	GET
http_status_code	String	Three-digit HTTP response code we returned	200
http_version	String	HTTP version (1.1 or 1.0)	HTTP/1.1
oauth_access_token	String	Detected access token (on OAuth 2.0 protected resources).	
package_name	String	Package name for which api_key is used	TestPackage
package_uuid	String	Unique identifier for the package	58c0b3cb-268e-4e92-bd08-1e38da4e1ac9
plan_name	String	Plan Name for the endpoint	Test plan
plan_uuid	String	Unique identifier for the plan	87116295-d119-41c0-88ba-da409ba2d760
pre_transfer_time	Number	Of the remote total_time, how long, in seconds, it took AFTER establishing a connection BEFORE we received the first byte.	0.001099
qps_throttle_	String	throttle_value	

Field	Type	Description	Example
value			
quota_value	Number	quota_value	
referrer	String	Referring URL for the call, if any	app:/MarketMeSuite.swf
remote_total_time	Number	Elapsed time, in seconds, the backend call took.	0.03747
request_host_name	String	Name of the host that took the call	xyz.abc.com
request_id	String	Local Edition Internal ID for this request	1306282895.4215_ bt6pnhuqhewew22xy7jmw7bar_ x84p868ucwkkvxwwtc7uzw5h
request_time	Number	Epoch microtime the request was received	1306282895.4215
request_uuid	String	Global unique identifier (GUID) of the call	26e2fa2d-3e63-4ec8-8aaa- fb4643e601cf
response_string	String	Response String having status and corresponding message	200 OK (API)
service_definition_endpoint_uuid	String	Unique identifier for the Service Endpoints	b6p4dgvhy2gjygcuvpqu6ekr
service_id	String	Spkey	kj94huqt2jqcz3qg7dvrsnmn
service_name	String	Name of service	testService

Field	Type	Description	Example
src_ip	String	Originating IP address of the call	1. 151.240.64
ssl_enabled	Number	Flag indicating whether or not SSL is enabled for the connection from the Traffic Manager to the Customer API	true
total_request_exec_time	Number	Elapsed time, in seconds, from the moment we got the request at the proxy worker to the moment we sent a response.	0.745138884
traffic_manager	String	Proxy worker that handled the request	ML5Test
traffic_manager_error_code	String	Internal error code for WHY we may have bounced the request. If '-', we didn't block the request from the endpoint.	ERR_403_DEVELOPER_INACTIVE
uri	String	Path+query string portion of the request	/TestAPI-34
user_agent	String	Application that made the request	curl/7.54.0

Location of Enriched Access Logs

/mnt/data/trafficmanager/enriched/access/access*.log

Sample output with headers

```
api_key,api_method_name,bytes,cache_hit,client_transfer_time,connect_time,endpoint_name,http_method,http_status_code,http_version,oauth_access_token,package_name,package_uuid,plan_name,plan_uuid,pre_transfer_time,qps_throttle_value,quota_value,referrer,remote_total_time,request_host_name,request_id,request_time,request_uuid,response_string,service_definition_endpoint_uuid,service_id,service_name,src_ip,ssl_enabled,total_request_exec_time,traffic_manager,traffic_manager_error_code,uri,user_agentsample,-,25,0,0.000,0.202,-,GET,200,HTTP/1.1,-,SamplePack,eebf7d77-53ed-43b2-bdbc-4d32270a6f57,SamplePackplanA,fc865c5f-2d78-4bc4-a0eb-ed8240e26557,0.001,1,1,-,0.211,1542643298.014_sample_gzbytfhwbxv393dectxdz8es,1542643298.014,e7ef2b0f-8a32-4fd2-a0bd-85bb155c5d9e,200OK(API),-,gzbytfhwbxv393dectxdz8es,-,127.0.0.1,0,16.388,tm-deploy-0-5d5d885577-xsg5m,-,/SampleAPI?api_key=sample,curl/7.29.0
```

Exporting Logs to Different Output Channels

The Log Service writes logs to a mounted persistent volume and are available at `/mnt/data`.



Note: Any property that begins with `tmg-internal` must not be changed.

Other than the default file destination, the Log Service can forward logs to Kafka, Database, Forward, Elasticsearch, HTTP endpoint, TCP socket, and SYSLOG server port. To forward to these destinations, use one of the recipes below and follow these steps:

1. Edit `<path-to-installer-directory>/tmgc-deploy/aws/k8s/properties/tml_log_properties.json`
2. Ensure that the hosts for any of the destinations are reachable via the network and that the required ports are not blocked for access.
3. Follow installation steps.

Properties for different output channels

The main property that needs to be set is the output channel marked by `td_agent_output_channelType`, `td_agent_metric_output_channelType`, `td_agent_container_output_channelType`, and `td_agent_verbose_output_channelType`.

The accepted values are **DEFAULT**, **DATABASE**, **FORWARD**, **ELASTICSEARCH**, **HTTP**, **KAFKA**, **TCP**, and **SYSLOG**. The values are case-sensitive.

i Note:

- These properties are defined in a JSON format and all JSON validation rules apply.
- All required values have to be provided, for example, even if destinations are using default ports, those have to be provided per recipe.
- All values must be set before starting the deployment.

Recipes

The Log Service can forward logs to:

- [File Destination](#)
- [Elasticsearch](#)
- [HTTP Endpoints](#)
- [Kafka](#)
- [TCP Socket](#)
- [SYSLOG](#)
- [MOM and HTTP](#)
- [Database](#)
- [Forward](#)

Recipe for File Destination

This destination does not have any additional properties.

tml_log_properties.json:

For access, metrics, verbose and component's logs to file based output channel.

```
{
  "tmg-internal-statusCheck": "OFF",
  "tmgc_log_service_logLevel": "info",
  "logserviceURL": "localhost",
  "td_agent_output_channelType": "DEFAULT",
  "td_agent_metric_output_channelType": "DEFAULT",
}
```

```
"td_agent_container_output_channelType": "DEFAULT",
"td_agent_verbose_output_channelType": "DEFAULT"
}
```

NOTE : This is the default output for all the types of logs.

- `td_agent_output_channelType` is for selecting output channel for access logs.
- `td_agent_metric_output_channelType` is for selecting output channel for metrics log.
- `td_agent_container_output_channelType` is for selecting output channel for component's logs. This includes applications, container agents and other short lived processes.
- `td_agent_verbose_output_channelType` is for selecting output channel for verbose logs.

Recipe for Elasticsearch

The Logging service requires certain properties written for Elasticsearch destinations. See the [Log Service Configuration Properties](#) table for valid values.

tml_log_properties.json

1. For access logs :

```
{
  "tmg-internal-statusCheck": "OFF",
  "tmgc_log_service_logLevel": "info",
  "logserviceURL": "localhost",
  "td_agent_metric_output_channelType": "DEFAULT",
  "td_agent_container_output_channelType": "DEFAULT",
  "td_agent_verbose_output_channelType": "DEFAULT",
  "td_agent_output_channelType": "ELASTICSEARCH",
  "td_agent_out_elasticsearch_host": "<network reachable elastic search host>",
  "td_agent_out_elasticsearch_port": "<valid ES port>",
  "td_agent_out_elasticsearch_index": "<elastic search index to update for new entries>",
  "td_agent_out_elasticsearch_user": "user", "td_agent_out_elasticsearch_password": "xxxxxx",
  "td_agent_out_elasticsearch_protocol": "https",
  "td_agent_out_elasticsearch_ssl_version": "TLSv1_2",
  "td_agent_out_elasticsearch_cafile": "<valid certificate name>"
}
```

2. For metrics logs :

```
{
  "tmg-internal-statusCheck": "OFF",
```

```

"tmgc_log_service_logLevel": "info",
"logserviceURL": "localhost",
"td_agent_container_output_channelType": "DEFAULT",
"td_agent_output_channelType": "DEFAULT",
"td_agent_verbose_output_channelType": "DEFAULT",
"td_agent_metric_output_channelType": "ELASTICSEARCH",
"td_agent_out_metric_elasticsearch_host": "<network reachable elastic search host>",
"td_agent_out_metric_elasticsearch_port": "<valid ES port>",
"td_agent_out_metric_elasticsearch_index": "<elastic search index to update for new entries>",
"td_agent_out_metric_elasticsearch_user": "user", "td_agent_out_metric_elasticsearch_
password": "xxxxxx",
"td_agent_out_metric_elasticsearch_protocol": "https",
"td_agent_out_metric_elasticsearch_ssl_version": "TLSv1_2",
"td_agent_out_metric_elasticsearch_cafire": "<valid certificate name>"
}

```

3. For Component/Container's logs :

```

{
  "tmg-internal-statusCheck": "OFF",
  "tmgc_log_service_logLevel": "info",
  "logserviceURL": "localhost",
  "td_agent_output_channelType": "DEFAULT",
  "td_agent_metric_output_channelType": "DEFAULT",
  "td_agent_verbose_output_channelType": "DEFAULT",
  "td_agent_container_output_channelType": "ELASTICSEARCH",
  "td_agent_out_container_elasticsearch_host": "<network reachable elastic search host>",
  "td_agent_out_container_elasticsearch_port": "<valid ES port>",
  "td_agent_out_container_elasticsearch_index": "<elastic search index to update for new
entries>",
  "td_agent_out_container_elasticsearch_user": "user", "td_agent_out_container_elasticsearch_
password": "xxxxxx",
  "td_agent_out_container_elasticsearch_protocol": "https",
  "td_agent_out_container_elasticsearch_ssl_version": "TLSv1_2",
  "td_agent_out_container_elasticsearch_cafire": "<valid certificate name>"
}

```

4. For verbose logs :

```

{
  "tmg-internal-statusCheck": "OFF",
  "tmgc_log_service_logLevel": "info",
  "logserviceURL": "localhost",
  "td_agent_output_channelType": "DEFAULT",
  "td_agent_metric_output_channelType": "DEFAULT",
  "td_agent_container_output_channelType": "DEFAULT",

```

```

"td_agent_verbose_output_channelType": "ELASTICSEARCH",
"td_agent_out_verbose_elasticsearch_host": "<network reachable elastic search host>",
"td_agent_out_verbose_elasticsearch_port": "<valid ES port>",
"td_agent_out_verbose_elasticsearch_index": "<elastic search index to update for new entries>",
"td_agent_out_verbose_elasticsearch_user": "user",
"td_agent_out_verbose_elasticsearch_password": "xxxxxx",
"td_agent_out_verbose_elasticsearch_protocol": "https",
"td_agent_out_verbose_elasticsearch_ssl_version": "TLSv1_2",
"td_agent_out_verbose_elasticsearch_cafile": "<valid certificate name>"
}

```

NOTE :

User can send all the types of logs to same elastic search, but index should be different in order to distinguish between different types of logs.

e.g.

```

{
  "tmg-internal-statusCheck": "OFF",
  "tmgc_log_service_logLevel": "info",
  "logserviceURL": "localhost",
  "td_agent_container_output_channelType": "ELASTICSEARCH",
  "td_agent_out_container_elasticsearch_host": "<network reachable elastic search host>",
  "td_agent_out_container_elasticsearch_port": "<valid ES port>",
  "td_agent_out_container_elasticsearch_index": "container",
  "td_agent_metric_output_channelType": "ELASTICSEARCH",
  "td_agent_out_metric_elasticsearch_host": "<network reachable elastic search host>",
  "td_agent_out_metric_elasticsearch_port": "<valid ES port>",
  "td_agent_out_metric_elasticsearch_index": "metrics",
  "td_agent_output_channelType": "ELASTICSEARCH",
  "td_agent_out_elasticsearch_host": "<network reachable elastic search host>",
  "td_agent_out_elasticsearch_port": "<valid ES port>",
  "td_agent_out_elasticsearch_index": "access",
  "td_agent_verbose_output_channelType": "ELASTICSEARCH",
  "td_agent_out_verbose_elasticsearch_host": "<network reachable elastic search host>",
  "td_agent_out_verbose_elasticsearch_port": "<valid ES port>",
  "td_agent_out_verbose_elasticsearch_index": "verbose"
}

```

Recipe for HTTP Endpoints

See the [Log Service](#) Configuration Properties table for valid values.

tml_log_properties.json

1. For access logs :

```
{
  "tmg-internal-statusCheck":"OFF",
  "tmgc_log_service_logLevel" : "info",
  "logserviceURL" : "localhost",
  "td_agent_container_output_channelType" : "DEFAULT",
  "td_agent_metric_output_channelType" : "DEFAULT",
  "td_agent_verbose_output_channelType" : "DEFAULT",
  "td_agent_output_channelType" : "HTTP",
  "td_agent_out_http_URI" : "<network reachable HTTP endpoint>"
}
```

2. For Component/Container's logs :

```
{
  "tmg-internal-statusCheck":"OFF",
  "tmgc_log_service_logLevel" : "info",
  "logserviceURL" : "localhost",
  "td_agent_output_channelType" : "DEFAULT",
  "td_agent_metric_output_channelType" : "DEFAULT",
  "td_agent_verbose_output_channelType" : "DEFAULT",
  "td_agent_container_output_channelType" : "HTTP",
  "td_agent_out_container_http_URI" : "<network reachable HTTP endpoint>"
}
```

3. For metrics logs :

```
{
  "tmg-internal-statusCheck":"OFF",
  "tmgc_log_service_logLevel" : "info",
  "logserviceURL" : "localhost",
  "td_agent_output_channelType" : "DEFAULT",
  "td_agent_container_output_channelType" : "DEFAULT",
  "td_agent_verbose_output_channelType" : "DEFAULT",
  "td_agent_metric_output_channelType" : "HTTP",
  "td_agent_out_metric_http_URI" : "<network reachable HTTP endpoint>"
}
```

4. For verbose logs :

```
{
  "tmg-internal-statusCheck":"OFF",
  "tmgc_log_service_logLevel" : "info",
  "logserviceURL" : "localhost",
```

```

"td_agent_output_channelType": "DEFAULT",
"td_agent_container_output_channelType": "DEFAULT",
"td_agent_metric_output_channelType": "DEFAULT",
"td_agent_verbose_output_channelType": "HTTP",
"td_agent_out_verbose_http_URI": "<network reachable HTTP endpoint>"
}

```

NOTE :

User can send all the types of logs to same destination over http
e.g.

```

{
  "tmg-internal-statusCheck": "OFF",
  "tmgc_log_service_logLevel": "info",
  "logserviceURL": "localhost",
  "td_agent_output_channelType": "HTTP",
  "td_agent_out_http_URI": "<network reachable HTTP endpoint>",
  "td_agent_metric_output_channelType": "HTTP",
  "td_agent_out_metric_http_URI": "<network reachable HTTP endpoint>",
  "td_agent_container_output_channelType": "HTTP",
  "td_agent_out_container_http_URI": "<network reachable HTTP endpoint>",
  "td_agent_verbose_output_channelType": "HTTP",
  "td_agent_out_verbose_http_URI": "<network reachable HTTP endpoint>"
}

```

Recipe for Kafka

See the [Log Service](#) Configuration Properties table for valid values.

tml_log_properties.json

1. For access logs :

```

{
  "tmg-internal-statusCheck": "OFF",
  "tmgc_log_service_logLevel": "info",
  "logserviceURL": "localhost",
  "td_agent_container_output_channelType": "DEFAULT",
  "td_agent_metric_output_channelType": "DEFAULT",
  "td_agent_verbose_output_channelType": "DEFAULT",
  "td_agent_output_channelType": "KAFKA",
  "td_agent_out_kafka_brokers": "<network reachable comma separated list of kafka broker:port>",
  "td_agent_out_kafka_topic": "<Kafka topic to publish to>"
}

```

2. For Metrics logs :

```
{
  "tmg-internal-statusCheck":"OFF",
  "tmgc_log_service_logLevel" : "info",
  "logserviceURL" : "localhost",
  "td_agent_output_channelType" : "DEFAULT",
  "td_agent_container_output_channelType" : "DEFAULT",
  "td_agent_verbose_output_channelType" : "DEFAULT",
  "td_agent_metric_output_channelType" : "KAFKA",
  "td_agent_out_metric_kafka_brokers" : "<network reachable comma separated list of kafka broker:port>",
  "td_agent_out_metric_kafka_topic" : "<Kafka topic to publish to>"
}
```

3. For Container/Component's logs :

```
{
  "tmg-internal-statusCheck":"OFF",
  "tmgc_log_service_logLevel" : "info",
  "logserviceURL" : "localhost",
  "td_agent_output_channelType" : "DEFAULT",
  "td_agent_metric_output_channelType" : "DEFAULT",
  "td_agent_verbose_output_channelType" : "DEFAULT",
  "td_agent_container_output_channelType" : "KAFKA",
  "td_agent_out_container_kafka_brokers" : "<network reachable comma separated list of kafka broker:port>",
  "td_agent_out_container_kafka_topic" : "<Kafka topic to publish to>"
}
```

4. For verbose logs :

```
{
  "tmg-internal-statusCheck":"OFF",
  "tmgc_log_service_logLevel" : "info",
  "logserviceURL" : "localhost",
  "td_agent_output_channelType" : "DEFAULT",
  "td_agent_metric_output_channelType" : "DEFAULT",
  "td_agent_container_output_channelType" : "DEFAULT",
  "td_agent_verbose_output_channelType" : "KAFKA",
  "td_agent_out_verbose_kafka_brokers" : "<network reachable comma separated list of kafka broker:port>",
  "td_agent_out_verbose_kafka_topic" : "<Kafka topic to publish to>"
}
```

NOTE :

User can send all the types of logs to same kafka brokers on different or same topic.

e.g.

```
{
  "tmg-internal-statusCheck":"OFF",
  "tmgc_log_service_logLevel" : "info",
  "logserviceURL" : "localhost",
  "td_agent_output_channelType" : "KAFKA",
  "td_agent_out_kafka_brokers" : "<network reachable comma separated list of kafka broker:port>",
  "td_agent_out_kafka_topic" : "access",
  "td_agent_container_output_channelType" : "KAFKA",
  "td_agent_out_container_kafka_brokers" : "<network reachable comma separated list of kafka broker:port>",
  "td_agent_out_container_kafka_topic" : "container",
  "td_agent_metric_output_channelType" : "KAFKA",
  "td_agent_out_metric_kafka_brokers" : "<network reachable comma separated list of kafka broker:port>",
  "td_agent_out_metric_kafka_topic" : "metric",
  "td_agent_verbose_output_channelType" : "KAFKA",
  "td_agent_out_verbose_kafka_brokers" : "<network reachable comma separated list of kafka broker:port>",
  "td_agent_out_verbose_kafka_topic" : "<Kafka topic to publish to>"
}
```

Recipe for TCP Socket

See the [Log Service](#) Configuration Properties table for valid values.

tml_log_properties.json

1. For access logs :

```
{
  "tmg-internal-statusCheck":"OFF",
  "tmgc_log_service_logLevel" : "info",
  "logserviceURL" : "localhost",
  "td_agent_metric_output_channelType" : "DEFAULT",
  "td_agent_container_output_channelType" : "DEFAULT",
  "td_agent_verbose_output_channelType" : "DEFAULT",
  "td_agent_output_channelType" : "TCP",
  "td_agent_out_tcp_host" : "<network reachable hostname or IP address>",
  "td_agent_out_tcp_port" : "<valid TCP port number>"
}
```

```
}
```

2. For metrics logs :

```
{
  "tmg-internal-statusCheck":"OFF",
  "tmgc_log_service_logLevel" : "info",
  "logserviceURL" : "localhost",
  "td_agent_output_channelType" : "DEFAULT",
  "td_agent_container_output_channelType" : "DEFAULT",
  "td_agent_verbose_output_channelType" : "DEFAULT",
  "td_agent_metric_output_channelType" : "TCP",
  "td_agent_out_metric_tcp_host" : "<network reachable hostname or IP address>",
  "td_agent_out_metric_tcp_port" : "<valid TCP port number>"
}
```

3. For container/component's logs :

```
{
  "tmg-internal-statusCheck":"OFF",
  "tmgc_log_service_logLevel" : "info",
  "logserviceURL" : "localhost",
  "td_agent_metric_output_channelType" : "DEFAULT",
  "td_agent_output_channelType" : "DEFAULT",
  "td_agent_verbose_output_channelType" : "DEFAULT",
  "td_agent_container_output_channelType" : "TCP",
  "td_agent_verbose_output_channelType" : "DEFAULT",
  "td_agent_out_container_tcp_host" : "<network reachable hostname or IP address>",
  "td_agent_out_container_tcp_port" : "<valid TCP port number>"
}
```

4. For verbose logs :

```
{
  "tmg-internal-statusCheck":"OFF",
  "tmgc_log_service_logLevel" : "info",
  "logserviceURL" : "localhost",
  "td_agent_metric_output_channelType" : "DEFAULT",
  "td_agent_output_channelType" : "DEFAULT",
  "td_agent_container_output_channelType" : "DEFAULT",
  "td_agent_verbose_output_channelType" : "TCP",
  "td_agent_out_verbose_tcp_host" : "<network reachable hostname or IP address>",
  "td_agent_out_verbose_tcp_port" : "<valid TCP port number>"
}
```

NOTE :

User can send all the types of logs to same destination over tcp

```
{
  "tmg-internal-statusCheck":"OFF",
  "tmgc_log_service_logLevel": "info",
  "logserviceURL": "localhost",
  "td_agent_output_channelType": "TCP",
  "td_agent_out_tcp_host": "<network reachable hostname or IP address>",
  "td_agent_out_tcp_port": "<valid TCP port number>",
  "td_agent_metric_output_channelType": "TCP",
  "td_agent_out_metric_tcp_host": "<network reachable hostname or IP address>",
  "td_agent_out_metric_tcp_port": "<valid TCP port number>",
  "td_agent_container_output_channelType": "TCP",
  "td_agent_out_container_tcp_host": "<network reachable hostname or IP address>",
  "td_agent_out_container_tcp_port": "<valid TCP port number>",
  "td_agent_verbose_output_channelType": "TCP",
  "td_agent_out_verbose_tcp_host": "<network reachable hostname or IP address>",
  "td_agent_out_verbose_tcp_port": "<valid TCP port number>"
}
```

Recipe for SYSLOG Socket

See the [Log Service](#) Configuration Properties table for valid values.

i Note: The format of the RFC message is RFC 3164. For more information, see [RFC 3164](#)

tml_log_properties.json

1. For access logs :

```
{
  "tmg-internal-statusCheck":"OFF",
  "tmgc_log_service_logLevel": "info",
  "logserviceURL": "localhost",
  "td_agent_metric_output_channelType": "DEFAULT",
  "td_agent_container_output_channelType": "DEFAULT",
  "td_agent_verbose_output_channelType": "DEFAULT",
  "td_agent_output_channelType": "SYSLOG",
  "td_agent_out_syslog_host": "<network reachable hostname or IP address syslog server>",
  "td_agent_out_syslog_port": "<valid syslog server port number>",
  "td_agent_out_syslog_tag": "ml5_accesslogs",
}
```

```
"td_agent_out_syslog_protocol":"tcp"
}
```

2. For metrics logs :

```
{
  "tmg-internal-statusCheck":"OFF",
  "tmgc_log_service_logLevel" : "info",
  "logserviceURL" : "localhost",
  "td_agent_output_channelType" : "DEFAULT",
  "td_agent_container_output_channelType" : "DEFAULT",
  "td_agent_verbose_output_channelType" : "DEFAULT",
  "td_agent_metric_output_channelType" : "SYSLOG",
  "td_agent_out_metric_syslog_host" : "<network reachable hostname or IP address syslog server>",
  "td_agent_out_metric_syslog_port" : "<valid syslog server port number>",
  "td_agent_out_metric_syslog_tag":"ml5_metriclogs",
  "td_agent_out_metric_syslog_protocol":"tcp"
}
```

3. For container/component's logs :

```
{
  "tmg-internal-statusCheck":"OFF",
  "tmgc_log_service_logLevel" : "info",
  "logserviceURL" : "localhost",
  "td_agent_metric_output_channelType" : "DEFAULT",
  "td_agent_output_channelType" : "DEFAULT",
  "td_agent_verbose_output_channelType" : "DEFAULT",
  "td_agent_container_output_channelType" : "SYSLOG",
  "td_agent_out_container_syslog_host" : ""<network reachable hostname or IP address syslog server>",
  "td_agent_out_container_syslog_port" : "<valid syslog server port number>",
  "td_agent_out_container_syslog_tag":"ml5_containerlogs",
  "td_agent_out_container_syslog_protocol":"tcp"
}
```

4. For verbose logs :

```
{
  "tmg-internal-statusCheck":"OFF",
  "tmgc_log_service_logLevel" : "info",
  "logserviceURL" : "localhost",
  "td_agent_metric_output_channelType" : "DEFAULT",
  "td_agent_output_channelType" : "DEFAULT",
  "td_agent_container_output_channelType" : "DEFAULT",
}
```

```

"td_agent_verbose_output_channelType": "SYSLOG",
"td_agent_out_verbose_syslog_host": ""<network reachable hostname or IP address syslog
server>",
"td_agent_out_verbose_syslog_port": "<valid syslog server port number>",
"td_agent_out_verbose_syslog_tag": "ml5_verboselogs",
"td_agent_out_verbose_syslog_protocol": "tcp"
}

```

NOTE :

User can send all the types of logs to same destination over syslog

```

{
"td_agent_output_channelType": "DEFAULT,SYSLOG",
"td_agent_container_output_channelType": "DEFAULT,SYSLOG",
"td_agent_metric_output_channelType": "DEFAULT,SYSLOG",
"td_agent_verbose_output_channelType": "DEFAULT,SYSLOG"
"td_agent_out_syslog_host": "<SYSLOG HOST>",
"td_agent_out_syslog_port": "5142",
"td_agent_out_syslog_tag": "ml5_accesslogs",
"td_agent_out_syslog_protocol": "tcp",
"td_agent_out_metric_syslog_host": "<SYSLOG HOST>",
"td_agent_out_metric_syslog_port": "5142",
"td_agent_out_metric_syslog_tag": "ml5_metriclogs",
"td_agent_out_metric_syslog_protocol": "tcp",
"td_agent_out_container_syslog_host": "<SYSLOG Host>",
"td_agent_out_container_syslog_port": "5142",
"td_agent_out_container_syslog_tag": "ml5_containerlogs",
"td_agent_out_container_syslog_protocol": "tcp"
"td_agent_out_verbose_syslog_host": ""<network reachable hostname or IP address syslog
server>",
"td_agent_out_verbose_syslog_port": "<valid syslog server port number>",
"td_agent_out_verbose_syslog_tag": "ml5_verboselogs",
"td_agent_out_verbose_syslog_protocol": "tcp"
}

```

Recipe for MOM and HTTP

See the [Log Service](#) Configuration Properties table for valid values.

The recipe for exporting logs to MOM and other output channels is a two-stage process.

1. Input MOM details in `tml_cluster_properties.json`. See the following sample:


```
{
  "mysql_root_pwd": "changeme",
  "mysql_masheryonprem_pwd": "changeme",
  "mysql_mashonpremrepl_pwd": "changeme",
  "mysql_mashclient_pwd": "changeme",
  "mysql_masherybackup_pwd": "changeme",
  "mom_server": "https://api-mom.mashery.com",
  "mom_key": "XXXXXXXXXXXXXXXXXXXXXXXXXXXX",
  "mom_secret": "xxxxxxxxx"
}
```

2. Input data for other output channels in `tml_log_properties.json`. See the following sample:

```
{
  "auto_binding": "ON",
  "td_agent_output_channelType": "DEFAULT,HTTP",
  "td_agent_container_output_channelType": "DEFAULT",
  "td_agent_metric_output_channelType": "DEFAULT",
  "td_agent_verbose_output_channelType": "DEFAULT",
  "td_agent_out_http_URI" : "http://nn.nn.nnn.nnn:nnnn/logs/accesslog"
}
```

Configuring Local Edition to Send Logs to Splunk

You can configure Boomi Cloud™ API Management - Local Edition with Splunk to monitor log information.

Complete the following steps to send logs to Splunk.

1. Configure the log pod/container to send access logs to a separate fluentd or td-agent server using Forward output. Refer the access logs of [Recipe for Forward](#).
2. On the external fluentd or td-agent server install the output plugin.
For more information see [fluentd documents](#) and [GitHub](#).
3. After setting up fluentd, edit the `fluent.conf` content at `/etc/fluent/fluent.conf` with the following content.

```

<system>
# trace equal to -vv option
# debug equal to -v option
log_level info
#change the worker count as per requirement
workers 5
</system>

<worker 0-4>
<source>
@type forward
port 24224
</source>
<match tml-enriched-logs>
@type splunkhec
host <splunk host_name>
port <port>
token 00000000-0000-0000-0000-000000000000

# metadata parameter
default_source tml-access-log
default_index tml-access-log
</match>
</worker>

```

i Note: You can edit the settings as per [information available on GitHub](#) and can also use the other configurable parameters given on this page.

4. Restart the fluentd service.

Splunk now can be used to see data on index that is provided in the above fluentd's configuration.

Recipe for Database

The Log Service can forward logs for the following database:

- MySQL server version greater than 5.5.
- PostgreSQL server version greater than 9.2.

Preparing MySQL

1. Create database users as required.
2. Create schema and required tables.

Note: To create the required database and tables, refer to `mlaccesslogschema-mysql.sql` provided in the `tml-spotfire` archive.

3. Provide permissions to the users for the newly created database and table.
4. Enable remote access to schema for the new users.

On running the SQL, you will see a database named `masherylocallogs` and a table named `tmlaccesslog`.

Note: To see the changes, use command-line interface or database client.

Preparing PostgreSQL

- The process for **Preparing PostgreSQL** is same as that of **Preparing MySQL**, explained above.
- To create the required database and tables, refer `mlaccesslogschema-postgres.sql` provided in the `tml-spotfire` archive.

Configuring Boomi Cloud™ API Management - Local Edition Log service to send access logs to the database

See the [Log Service](#) Configuration Properties table for valid values.

`tml_log_properties.json`

```
{
  "tmg-internal-statusCheck": "OFF",
  "tmgc_log_service_logLevel": "info",
  "logserviceURL": "localhost",
  "td_agent_output_channelType": "DATABASE",
  "td_agent_out_dbType": "<MYSQL or POSTGRES>",
  "td_agent_out_dbHost": "<network reachable database host>",
  "td_agent_out_dbPort": "<port for database>",
  "td_agent_out_dbSchema": "<name of the database>",
```

```
"td_agent_out_dbUser" : "<db_user>",
"td_agent_out_dbPassword" : "<db_password>"
}
```

Recipe for Forward

See the [Log Service](#) Configuration Properties table for valid values.

tml_log_properties.json

- For access logs:

```
{
  "tmg-internal-statusCheck": "OFF",
  "tmgc_log_service_logLevel" : "info",
  "logserviceURL" : "localhost",
  "td_agent_output_channelType" : "FORWARD",
  "td_agent_out_forward_servers" : "fluentHost1:PORT,fluentHost2:PORT"
}
```

- For metrics logs :

```
{
  "tmg-internal-statusCheck": "OFF",
  "tmgc_log_service_logLevel" : "info",
  "logserviceURL" : "localhost",
  "td_agent_metric_output_channelType" : "FORWARD",
  "td_agent_out_metric_forward_servers" : "fluentHost1:PORT,fluentHost2:PORT"
}
```

- For Container's logs :

```
{
  "tmg-internal-statusCheck": "OFF",
  "tmgc_log_service_logLevel" : "info",
  "logserviceURL" : "localhost",
  "td_agent_container_output_channelType" : "FORWARD",
  "td_agent_out_container_forward_servers" : "fluentHost1:PORT,fluentHost2:PORT"
}
```

- For Verbose logs :

```
{
  "tmg-internal-statusCheck":"OFF",
  "tmgc_log_service_logLevel" : "info",
  "logserviceURL" : "localhost",
  "td_agent_verbose_output_channelType" : "FORWARD",
  "td_agent_out_verbose_forward_servers" : "fluentHost1:PORT,fluentHost2:PORT"
}
```

- To send access logs, metrics logs, container's logs and verbose logs to the different FORWARD service, you must distinguish the different logs by providing different index values. For example:

```
{
  "tmg-internal-statusCheck":"OFF",
  "tmgc_log_service_logLevel" : "info",
  "logserviceURL" : "localhost",
  "td_agent_container_output_channelType" : "FORWARD",
  "td_agent_metric_output_channelType" : "FORWARD",
  "td_agent_verbose_output_channelType" : "FORWARD",
  "td_agent_output_channelType" : "FORWARD",
  "td_agent_out_container_forward_servers" : "fluentHost1:PORT,fluentHost2:PORT",
  "td_agent_out_metric_forward_servers" : "fluentHost1:PORT,fluentHost2:PORT",
  "td_agent_out_verbose_forward_servers" : "fluentHost1:PORT,fluentHost2:PORT"
}
```

Integrating with TIBCO Spotfire

Spotfire can consume data from a Local Edition database. Supported logs include the Traffic Manager Access logs.

When you extract the TIB_mash-local_5.*_cf.tar.gz file, you will see a tml-spotfire folder.

The tml-spotfire folder contains the following artifacts needed for the Local Edition-Spotfire integration.

Artifacts	Properties
Introduction to Mashery and Spotfire.dxp	Spotfire Analysis file for Local Edition logs.

Artifacts	Properties
mlaccesslogschema-mysql.sql	MySQL DB schema for Local Edition Log export.
mlaccesslogschema-postgres.sql	Postgres DB schema for Local Edition Log export.
mlaccesslogschema-mysql-v1.sql	MySQL scripts for upgrading from Local Edition 5.3 to Local Edition 5.3.1.
mlaccesslogschema-postgres-v1.sql	Postgres scripts for upgrading from Local Edition 5.3 to Local Edition 5.3.1.

For more information on upgrade and migration, see the section, "Upgrading Local Edition 5.3 to Local Edition 5.3.1" in *Boomi Cloud™ API Management - Local Edition Upgrade and Migration Guide*.

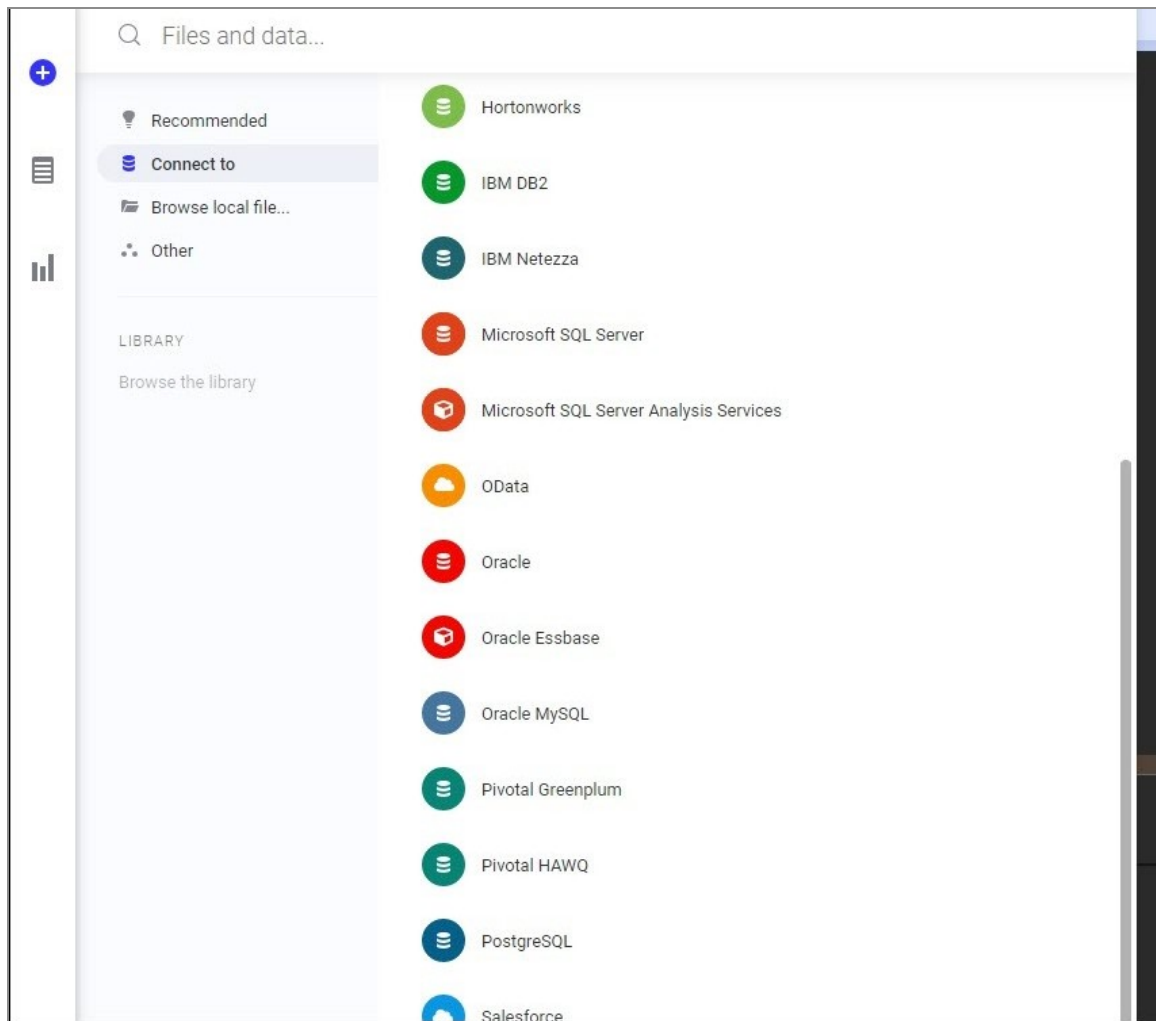
For information on forwarding logs to the database, refer [Recipe for Database](#).

Preparing TIBCO Spotfire to Consume Access Logs

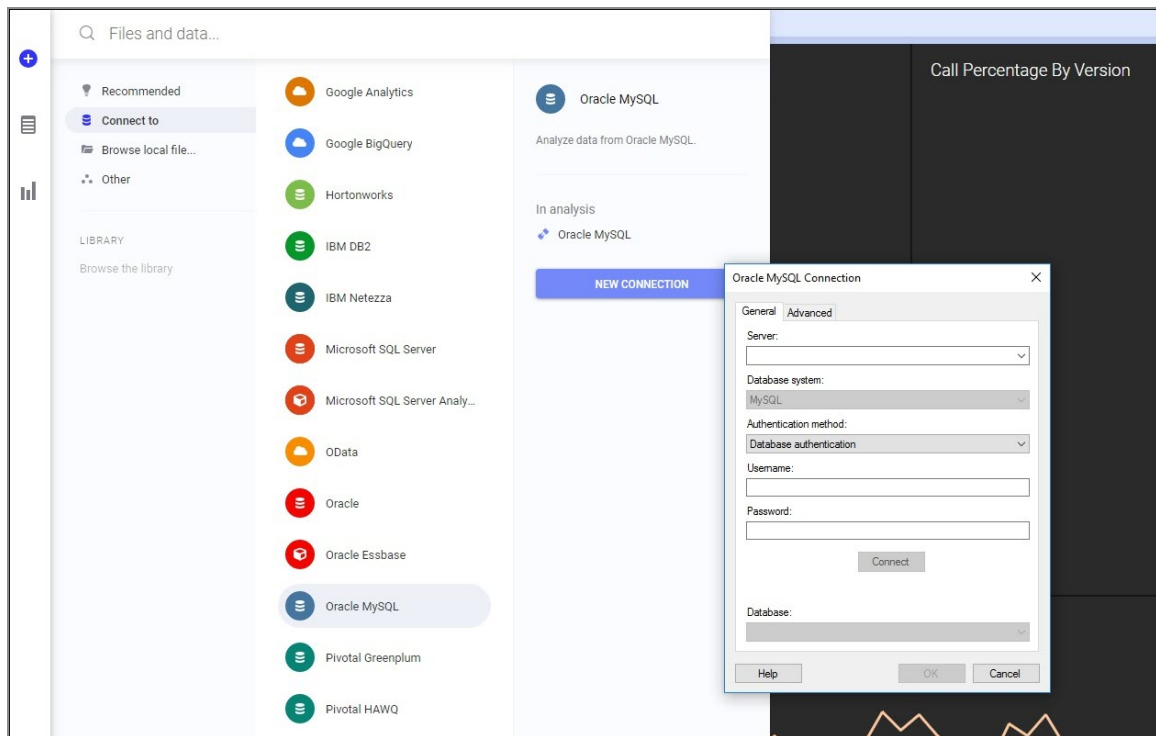
The following section describes how to prepare TIBCO Spotfire to consume access logs.

To prepare TIBCO Spotfire to consume access logs, perform the following steps:

1. Launch TIBCO Spotfire.
2. Open DXP.
3. Click the + button in upper-left navigation.
4. Click **Connect to**, select **Oracle MySQL**.



5. Select New Connection.



6. Specify the following information:

- **Server Name**
- **Username**
- **Password**

7. Click **Connect**.

8. Click **OK**.

Changing Output Destinations for Different Types of Logs using CLI

This topic contains information about how to configure output channels to send logs, access logs and metrics to Database, Forward, Elasticsearch, HTTP, Kafka, Syslog, generic TCP end points using command-line interface.

How to Use This Feature

This feature allows you to change the output destination for different types of logs in one invocation.



Caution: Using the CLI overwrites the existing configuration. Therefore, each time it is used, the system expects to receive a full configuration. Incomplete configurations are ignored when generating the log service's configurations.

The command accepts a JSON that contains the different configuration properties. The data in the file should adhere to JSON rules. All properties are top-level attributes. See the examples below.

Command Line

The generic form of the command is:

```
clustermanager import config --componentType <component type> [-- componentId <componentId>] --file  
<full path to json>
```

For example:

```
cm  
import  
config [--componentId 3a076250-955b-49ef-ab06-ea19df10a665 --componentType logservice --  
file props.json
```

Types of Logs

You can export the following types of logs to the chosen destination:

1. Access logs
2. Metrics
3. Container logs
4. Verbose logs

Each type of log has a different set of property names. This way you can send different types of logs to different destinations.

The logs are written to the disk (DEFAULT) on the log service container irrespective of the destination you choose.

Property names and description for each type of log

All values and property names are in character/string data type. Depending on chosen output channel type, the remaining configuration will change.

i Note: To configure log service to communicate with Elasticsearch via HTTPS, copy the CA certificate (in PEM form) of the elastic search host to /docker-deploy/resources/tml-log/. The files are in the log container.

Access Logs

Destination Type	Name in JSON	Description	Notes	Example
ELASTICSEARCH	td_agent_out_elasticsearch_hosts	Comma separated value of Elasticsearch host and port in the form host1:port1,host2:port2,host3:port3.	If both <i>td_agent_out_elasticsearch_hosts</i> and <i>td_agent_out_elasticsearch_host</i> are provided; only <i>td_agent_out_elasticsearch_hosts</i> will be considered. Ensure that each host are reachable from log pod.	"td_agent_out_elasticsearch_hosts":"host1:port1,host2:port2,host3:port3"
	td_agent_output_channelType	Comma separated	Applicable values are	"td_agent_output_channelType": "<Destination Type>"

Destination Type	Name in JSON	Description	Notes	Example
	output_channelType	string of types of channels	DEFAULT, DATABASE, FORWARD, ELASTICSEARCH, HTTP, KAFKA, SYSLOG, TCP.	
DATABASE	td_agent_out_dbType	Type of database to be connected.	Applicable values are MYSQL, POSTGRESQL	"td_agent_out_dbType":"MYSQL" OR "td_agent_out_dbType":"POSTGRESQL"
	td_agent_out_dbHost	DBHost where Database instance is running.	The hostname or IP should be reachable from the log service container	"td_agent_out_dbHost":"MYDBHOST" OR "td_agent_out_dbHost":"192.168.1.11"
	td_agent_out_dbPort	A port on which the database is listening.		"td_agent_out_dbPort":"3306" OR "td_agent_out_dbPort":"5432"
	td_agent_out_dbSchema	Database schema under which access_log table is created or existed.		"td_agent_out_dbSchema":"masherylocallogs"

Destination Type	Name in JSON	Description	Notes	Example
	td_agent_out_dbUser	The database user with permission to access.		"td_agent_out_dbUser":"DBUSER"
	td_agent_out_dbPassword	Password for given user.		"td_agent_out_dbPassword":"PASSWORD"
FORWARD	td_agent_out_forward_servers	Hostname or IP address and port for available FluentD servers. Expected value is [{HOST}:{PORT}, {HOST1}:{PORT}, {HOST2}:{PORT}]	The hostname should be reachable from the log service container. LogService service will push data to any of the available servers.	- With port: "td_agent_out_forward_servers": 192.168.1.11:24221, 192.168.2.11:24225" "td_agent_out_forward_servers" : fluentHost1:24221,fluentHost2:24225" - Without port: "td_agent_out_forward_servers" : fluentHost1,fluentHost2". Default port value: 24224
ELASTICSEARCH	td_agent_out_elasticsearch_host	Host name or IP address of Elastic search server or ELB.	The hostname should be reachable from within the log service container.	"td_agent_out_elasticsearch_host" : "a06ebe8a15a3011e9bf380608b2b6e73-531419666.us-east-1.elb.amazonaws.com"

Destination Type	Name in JSON	Description	Notes	Example
	td_agent_out_elasticsearch_port			"td_agent_out_elasticsearch_port" : "9200"
	td_agent_out_elasticsearch_index			"td_agent_out_elasticsearch_index" : "ml5_access_logs"
	td_agent_out_elasticsearch_protocol	The protocols supported are HTTP and HTTPS.	This is optional. The default protocol is HTTP.	"td_agent_out_elasticsearch_protocol": "https"
	td_agent_out_elasticsearch_ssl_version	SSL version configured on the destination elasticsearch instance.	Use this if td_agent_out_elasticsearch_protocol is HTTPS. Supported versions are TLSv1_2, TLSv1, TLSv1_1, SSLv23. Default version is TLSv1_2.	"td_agent_out_elasticsearch_ssl_version": "TLSv1_2"
	td_	The Name of	Use this if	"td_agent_out_elasticsearch_

Destination Type	Name in JSON	Description	Notes	Example
	agent_out_elasticsearch_arch_cafile	CA file to be used.	td_agent_out_elasticsearch_protocol is HTTPS. The value should be the name of the file.	cafile": "cafile-a06ebe8a15a3011e9bf380608b2-531419666.us-east-1.elb.amazonaws.com.crt"
	td_agent_out_elasticsearch_arch_user	User name to authenticate.	This is optional depending on the elastic search security settings.	"td_agent_out_elasticsearch_user": "elasticuser"
	td_agent_out_elasticsearch_arch_password	Credential		"td_agent_out_elasticsearch_password": "password123"
HTTP	td_agent_out_http_URI	HTTP URI to which logs are to be sent.	The HTTP URI should be accessible from the log container.	
KAFKA	td_agent_out_kafka_brokers	Comma delimited list of kafkabroker_host:port.	The Kafka brokers host should be reachable from the log service and broker	"td_agent_out_kafka_brokers": "broker1:9092,broker2:9092"

Destination Type	Name in JSON	Description	Notes	Example
			should be running.	
	td_agent_out_kafka_topic			"td_agent_out_kafka_topic": "ml5_access_logs"
SYSLOG	td_agent_out_syslog_host	Host name of IP address where syslog is running.	Host should be reachable from log service.	"td_agent_out_syslog_host": "a06ebe8a15a3011e9bf380608b2b6e73-531419666.us-east-1.elb.amazonaws.com"
	td_agent_out_syslog_port	Port number of syslog program.		"td_agent_out_syslog_port": "5142"
	td_agent_out_syslog_tag	Tag with which the log is to be uniquely identified.	Defaults to incoming log event tag.	"td_agent_out_syslog_tag": "ml5_access_logs"
	td_agent_out_syslog_protocol	Protocol on which syslog is listening for incoming requests.	Defaults to tcp.	"td_agent_out_syslog_protocol": "tcp"
TCP	td_agent_out_tcp_	Host name where a TCP socket is	Host should be reachable	"td_agent_out_tcp_host": "a06ebe8a15a3011e9bf380608b2b6e73-531419666.us-east-

Destination Type	Name in JSON	Description	Notes	Example
	host	listening.	from log service	1.elb.amazonaws.com"
	td_agent_out_tcp_port	Port on which the TCP socket is listening.		"td_agent_out_tcp_port":"6000"

Metrics Log

Destination Type	Name in JSON	Description	Notes	Example
ELASTICSEARCH	td_agent_out_elasticsearch_hosts	Comma separated value of Elasticsearch host and port in the form host1:port1,host2:port2,host3:port3.	If both <i>td_agent_out_elasticsearch_hosts</i> and <i>td_agent_out_elasticsearch_host</i> are provided; only <i>td_agent_out_elasticsearch_hosts</i> will be considered. Ensure that each host are reachable from log pod.	"td_agent_out_elasticsearch_hosts":"host1:port1,host2:port2,host3:port3"

Destination Type	Name in JSON	Description	Notes	Example
	td_agent_metric_output_channelType	Comma separated string of types of channels	Applicable values are FORWARD, ELASTICSEARCH, HTTP, KAFKA, SYSLOG, TCP	"td_agent_metric_output_channelType": "<Destination Type>"
FORWARD	td_agent_out_metric_forward_servers	Hostname or IP address and port for available FluentD servers Expected value is [{HOST}: {PORT}, {HOST1}: {PORT}: {HOST2}: {PORT}]	The hostname should be reachable from the log service container. LogService service will push data to any of the available servers.	- With port: "td_agent_out_metric_forward_servers": 192.168.1.11:24221, 192.168.2.11:24225" "td_agent_out_metric_forward_servers": fluentHost1:24221, fluentHost2:24225" - Without port: "td_agent_out_metric_forward_servers": fluentHost1, fluentHost2". Default port value: 24224
ELASTICSEARCH	td_agent_out_metric_elasticsearch_host	Host name or IP address of Elastic search server or ELB	The hostname should be reachable from within the log service container	"td_agent_out_metric_elasticsearch_host": "a06ebe8a15a3011e9bf380608b2b6e73-531419666.us-east-1.elb.amazonaws.com"

Destination Type	Name in JSON	Description	Notes	Example
	td_agent_out_metric_elasticsearch_port			"td_agent_out_metric_elasticsearch_port" : "9200"
	td_agent_out_metric_elasticsearch_index		Defaults to incoming log event tag.	"td_agent_out_metric_elasticsearch_index" : "ml5_metric_logs"
	td_agent_out_metric_elasticsearch_protocol	The protocols supported are HTTP and HTTPS.	This is optional. The default protocol is HTTP.	"td_agent_out_metric_elasticsearch_protocol": "https"
	td_agent_out_metric_elasticsearch_ssl_version	SSL version configured on the destination elasticsearch instance.	Use this if td_agent_out_elasticsearch_protocol is HTTPS. Supported versions are TLSv1_2, TLSv1, TLSv1_1, SSLv23. Default version is	"td_agent_out_metric_elasticsearch_ssl_version": "TLSv1_2"

Destination Type	Name in JSON	Description	Notes	Example
			TLSv1_2.	
	td_agent_out_metric_elasticsearch_cafile	The Name of CA file to be used.	Use this if td_agent_out_elasticsearch_protocol is HTTPS. The value should be the name of the file.	"td_agent_out_metric_elasticsearch_cafile": "cafile-a06ebe8a15a3011e9bf380608b2-531419666.us-east-1.elb.amazonaws.com.crt"
	td_agent_out_metric_elasticsearch_user	User name to authenticate.	This is optional depending on the elastic search security settings.	"td_agent_out_metric_elasticsearch_user": "elasticuser"
	td_agent_out_metric_elasticsearch_password	Credential		"td_agent_out_metric_elasticsearch_password": "password123"
HTTP	td_agent_out_metric_http_URI	HTTP URI to which logs are to be sent.	The HTTP URI should be accessible from the log container	"td_agent_out_metric_http_URI": "http://a06ebe8a15a3011e9bf380608b2b6e73-531419666.us-east-1.elb.amazonaws.com/ml5/metriclogs"
KAFKA	td_agent	Comma separated list	The Kafka brokers host	"td_agent_out_metric_kafka_brokers": "broker1:9092,broker2:909"

Destination Type	Name in JSON	Description	Notes	Example
	out_metric_kafka_brokers	of kafkabroker_host:port.	should be reachable from the log service and broker should be running.	2"
	td_agent_out_metric_kafka_topic			"td_agent_out_metric_kafka_topic": "ml5_metric_logs"
SYSLOG	td_agent_out_metric_syslog_host	Host name of IP address where syslog is running.	Host should be reachable from log service.	"td_agent_out_metric_syslog_host": "a06ebe8a15a3011e9bf380608b2b6e73-531419666.us-east-1.elb.amazonaws.com"
	td_agent_out_metric_syslog_port	Port number of syslog program.		"td_agent_out_metric_syslog_port": "5142"
	td_agent_out_metric_syslog_tag	Tag with which the log is to be uniquely identified.	Defaults to incoming log event tag.	"td_agent_out_metric_syslog_tag": "ml5_metric_logs"

Destination Type	Name in JSON	Description	Notes	Example
	td_agent_out_metric_syslog_protocol	Protocol on which syslog is listening for incoming requests.	Defaults to tcp.	"td_agent_out_metric_syslog_protocol":"tcp"
TCP	td_agent_out_metric_tcp_host	Host name where a TCP socket is listening.	Host should be reachable from log service.	"td_agent_out_metric_tcp_host": "a06ebe8a15a3011e9bf380608b2b6e73-531419666.us-east-1.elb.amazonaws.com"
	td_agent_out_metric_tcp_port	Port on which the TCP socket is listening.		"td_agent_out_metric_tcp_port":"6000"

Container Logs

Destination Type	Name in JSON	Description	Notes	Example
ELASTICSEARCH	td_agent_out_elasticsearch_hosts	Comma separated value of Elasticsearch host and port in the form host1:port1,host2:port2,host3:port3.	If both <i>td_agent_out_elasticsearch_hosts</i> and <i>td_agent_out_elasticsearch_host</i> are provided; only <i>td_agent_out_elasticsearch_hosts</i>	"td_agent_out_elasticsearch_hosts":"host1:port1,host2:port2,host3:port3"

Destination Type	Name in JSON	Description	Notes	Example
			<i>elasticsearch</i> <i>h_hosts</i> will be considered. Ensure that each host are reachable from log pod.	
	td_agent_container_output_channelType	Comma separated string of types of channels	Applicable ones are FORWARD, ELASTICSEARCH, HTTP, KAFKA, SYSLOG, TCP	"td_agent_container_output_channelType": "<Destination Type>"
FORWARD	td_agent_out_container_forward_servers	Hostname or IP address and port for available FluentD servers Expected value is [{HOST}: {PORT}, {HOST1}: {PORT}: {HOST2}: {PORT}]	The hostname should be reachable from the log service container. LogService service will push data to any of the available servers.	- With port: "td_agent_out_container_forward_servers": 192.168.1.11:24221, 192.168.2.11:24225" "td_agent_out_container_forward_servers" : fluentHost1:24221,fluentHost2:24225" - Without port: "td_agent_out_container_forward_servers" : fluentHost1,fluentHost2".

Destination Type	Name in JSON	Description	Notes	Example
ELASTICSEARCH				Default port value: 24224
	td_agent_out_container_r_elasticsearch_host		The hostname should be reachable from within the log service container	"td_agent_out_container_elasticsearch_host" : "a06ebe8a15a3011e9bf380608b2b6e73-531419666.us-east-1.elb.amazonaws.com"
	td_agent_out_container_r_elasticsearch_port			"td_agent_out_container_elasticsearch_port" : "9200"
	td_agent_out_container_r_elasticsearch_index		Defaults to incoming log event tag.	"td_agent_out_container_elasticsearch_index" : "ml5_container_logs"
	td_agent_out_container_r_elasticsearch_	The protocols supported are HTTP and HTTPS.	This is optional. The default protocol is HTTP.	"td_agent_out_container_elasticsearch_protocol": "https"

Destination Type	Name in JSON	Description	Notes	Example
	protocol			

Destination Type	Name in JSON	Description	Notes	Example
	td_agent_out_container_elasticsearch_ssl_version	SSL version configured on the destination elasticsearch instance.	Use this if td_agent_out_elasticsearch_protocol is HTTPS. Supported versions are TLSv1_2, TLSv1, TLSv1_1,SSLv23. Default version is TLSv1_2.	"td_agent_out_container_elasticsearch_ssl_version":"TLSv1_2"
	td_agent_out_container_elasticsearch_cafile	The Name of CA file to be used.	Use this if td_agent_out_elasticsearch_protocol is HTTPS. The value should be the name of the file.	" td_agent_out_container_elasticsearch_cafile":"cafile-a06ebe8a15a3011e9bf380608b2-531419666.us-east-1.elb.amazonaws.com.crt"
	td_agent_out_container_elasticsearch_user	User name to authenticate.	This is optional depending on the elastic search security settings.	"td_agent_out_container_elasticsearch_user":"elasticuser"
	td_agent_out_	Credential		" td_agent_out_container_elasticsearch_password":"password123"

Destination Type	Name in JSON	Description	Notes	Example
	container_elasticsearch_password			
HTTP	td_agent_out_container_http_URI	HTTP URI to which logs are to be sent.	The HTTP URI should be accessible from the log container.	"td_agent_out_container_http_URI": "http://a06ebe8a15a3011e9bf380608b2b6e73-531419666.us-east-1.elb.amazonaws.com/ml5/containerlogs"
KAFKA	td_agent_out_container_kafka_brokers	Comma separated list of kafkabroker_host:port.	The Kafka brokers host should be reachable from the log service and broker should be running.	"td_agent_out_container_kafka_brokers": "broker1:9092,broker2:9092"
	td_agent_out_container_kafka_topic		Defaults to incoming log event tag.	"td_agent_out_container_kafka_topic" : "ml5_container_logs"
SYSLOG	td_agent_out_container_syslog_host	Host name of IP address where syslog is running.	Host should be reachable from log service.	"td_agent_out_container_syslog_host": "a06ebe8a15a3011e9bf380608b2b6e73-531419666.us-east-1.elb.amazonaws.com"

Destination Type	Name in JSON	Description	Notes	Example
	td_agent_out_container_syslog_port	Port number of syslog program.		"td_agent_out_container_syslog_port": "5142"
	td_agent_out_container_syslog_tag	Tag with which the log is to be uniquely identified.	Defaults to incoming log event tag.	"td_agent_out_container_syslog_tag": "ml5_containerlogs"
	td_agent_out_container_syslog_protocol	Protocol on which syslog is listening for incoming requests.	Defaults to TCP.	"td_agent_out_container_syslog_protocol": "tcp"
TCP	td_agent_out_container_tcp_host	Host name where a TCP socket is listening.	Host should be reachable from log service.	"td_agent_out_container_tcp_host": "a06ebe8a15a3011e9bf380608b2b6e73-531419666.us-east-1.elb.amazonaws.com"
	td_agent_out_container_tcp_port	Port on which the TCP socket is listening.		"td_agent_out_container_tcp_port": "6000"

Verbose Logs

Destination Type	Name in JSON	Description	Notes	Example
	td_agent_verbose_output_channelType	Comma separated string of types of channels	Applicable ones are FORWARD, ELASTICSEARCH, HTTP, KAFKA, SYSLOG, TCP	"td_agent_verbose_output_channelType": "<Destination Type>"
FORWARD	td_agent_out_verbose_forward_servers	Hostname or IP address and port for available FluentD servers Expected value is [{HOST}:{PORT}, {HOST1}:{PORT}, {HOST2}:{PORT}]	The hostname should be reachable from the log service container. LogService service will push data to any of the available servers.	- With port: "td_agent_out_verbose_forward_servers": 192.168.1.11:24221,192.168.2.11:24225" "td_agent_out_verbose_forward_servers" : fluentHost1:24221,fluentHost2:24225" - Without port: "td_agent_out_verbose_forward_servers" : fluentHost1,fluentHost2". Default port value: 24224
ELASTICSEARCH	td_agent_out_verbose_elasticsearch_host		The hostname should be reachable from within the log service container	"td_agent_out_verbose_elasticsearch_host" : "a06ebe8a15a3011e9bf380608b2b6e73-531419666.us-east-1.elb.amazonaws.com"

Destination Type	Name in JSON	Description	Notes	Example
	td_agent_out_verbose_elasticsearch_port			"td_agent_out_verbose_elasticsearch_port" : "9200"
	td_agent_out_verbose_elasticsearch_index		Defaults to incoming log event tag.	"td_agent_out_verbose_elasticsearch_index" : "ml5_container_logs"
	td_agent_out_verbose_elasticsearch_protocol	The protocols supported are HTTP and HTTPS.	This is optional. The default protocol is HTTP.	"td_agent_out_verbose_elasticsearch_protocol": "https"
	td_agent_out_verbose_elasticsearch_ssl_version	SSL version configured on the destination Elasticsearch instance.	Use this if td_agent_out_elasticsearch_protocol is HTTPS. Supported versions are TLSv1_2, TLSv1, TLSv1_1, SSLv23. Default version is TLSv1_2.	"td_agent_out_verbose_elasticsearch_ssl_version": "TLSv1_2"
	td_agent_out_verbose_elasticsearch_index	The	Use this if td_	" td_agent_out_verbose_

Destination Type	Name in JSON	Description	Notes	Example
	out_verbose_elasticsearch_cafile	Name of CA file to be used.	agent_out_elasticsearch_protocol is HTTPS. The value should be the name of the file.	elasticsearch_cafile": "cafile-a06ebe8a15a3011e9bf380608b2-531419666.us-east-1.elb.amazonaws.com.crt"
	td_agent_out_verbose_elasticsearch_user	User name to authenticate.	This is optional depending on the elastic search security settings.	"td_agent_out_verbose_elasticsearch_user": "elasticuser"
	td_agent_out_verbose_elasticsearch_password	Credential		"td_agent_out_verbose_elasticsearch_password": "password123"
HTTP	td_agent_out_verbose_http_URI	HTTP URI to which logs are to be sent.	The HTTP URI should be accessible from the log container.	"td_agent_out_verbose_container_http_URI": "http://a06ebe8a15a3011e9bf380608b2b6e73-531419666.us-east-1.elb.amazonaws.com/ml5/containerlogs"
KAFKA	td_agent_out_verbose_kafka_brokers	Comma separated list of kafka broker host:port.	The Kafka brokers host should be reachable from the log service and broker should be running.	"td_agent_out_verbose_kafka_brokers": "broker1:9092,broker2:9092"

Destination Type	Name in JSON	Description	Notes	Example
	td_agent_out_verbose_kafka_topic		Defaults to incoming log event tag.	"td_agent_out_verbose_kafka_topic" : "ml5_container_logs"
SYSLOG	td_agent_out_verbose_syslog_host	Host name of IP address where syslog is running.	Host should be reachable from log service.	"td_agent_out_verbose_syslog_host": "a06ebe8a15a3011e9bf380608b2b6e73-531419666.us-east-1.elb.amazonaws.com"
	td_agent_out_verbose_syslog_port	Port number of syslog program.		"td_agent_out_verbose_syslog_port": "5142"
	td_agent_out_verbose_syslog_tag	Tag with which the log is to be uniquely identified.	Defaults to incoming log event tag.	"td_agent_out_verbose_syslog_tag": "ml5_containerlogs"
	td_agent_out_verbose_syslog_protocol	Protocol on which syslog is listening for incoming requests.	Defaults to TCP.	"td_agent_out_verbose_syslog_protocol": "tcp"

Destination Type	Name in JSON	Description	Notes	Example
TCP	td_agent_out_verbose_tcp_host	Host name where a TCP socket is listening.	Host should be reachable from log service.	"td_agent_out_verbose_tcp_host": "a06ebe8a15a3011e9bf380608b2b6e73-531419666.us-east-1.elb.amazonaws.com"
	td_agent_out_verbose_tcp_port	Port on which the TCP socket is listening.		"td_agent_out_verbose_tcp_port": "6000"

Importing Configurations From Cluster Manager Command Line Interface

To import configurations from cluster manager command line interface follow these steps:

1. List components:

```

clustermanager ls components
Using cluster [tml52clusterpune1]
Using Zone [eastus]
Component ID      Type      Name      Status  Last Heartbeat Received
Host      Service Port(s)
-----
f41c95b9-6059-42c1-8a4f-fb2261974c14 cache      cache-set-0-0.cac... ACTIVE Mar 12
2020 08:59:09 +0000 10.244.1.6 11212,11211,11213,11214,11215,11216
710b9df4-399a-4551-b1ed-c1fc1f8ed744 configmanager cm-deploy-0-85bf... ACTIVE
Mar 12 2020 08:59:25 +0000 10.244.2.6 8080
c1057ef0-ab4e-459f-8413-0e5036c65b7c logservice log-set-0-0.log-s... ACTIVE Mar 12
2020 08:59:30 +0000 10.244.2.7 24224
d516fe9e-3ac3-4a09-a9b0-1dceba36a23d nosql      cass-set-0-2.cass... ACTIVE Mar 12
2020 08:59:09 +0000 10.244.1.5 9042
29812c89-2e87-4dc0-a981-890cd1a72933 nosql      cass-set-0-0.cass... ACTIVE Mar 12

```



```

2020 08:59:09 +0000 10.244.2.5 9042
99950720-6ded-46be-a87f-dc82be47f265 nosql cass-set-0-1.cass... ACTIVE Mar 12
2020 08:59:09 +0000 10.244.0.6 9042
8a5a78f0-f032-4046-ba59-28a75004c9ce sql mysql-set-0-0.mys... ACTIVE Mar 12
2020 08:59:34 +0000 10.244.2.8 3306
111628dc-7893-45c8-8add-8a7944a2ff8c trafficmanager tm-deploy-0-5f787... ACTIVE
Mar 12 2020 08:59:28 +0000 10.244.0.7 8080

```

2. Get the component ID of **logservice** from the output.
3. Check the existing configuration.

```

clustermanager ls config --componentId c1057ef0-ab4e-459f-8413-0e5036c65b7c --
componentType logservice
clusterId [ad41097e-35bc-4fc4-bb22-084dc25fdbf2] and zoneId [9046178a-00f9-4d59-a92e-
59824142df56]
No settings found for the given componentId [c1057ef0-ab4e-459f-8413-0e5036c65b7c] of
type [logservice]

```

4. Create a JSON file for input (for example, /home/builder/property.json).
5. Update the configuration by importing the JSON file.

```

clustermanager import config --componentId c1057ef0-ab4e-459f-8413-0e5036c65b7c --
componentType logservice --file /home/builder/property.json
Updating the TMGC after the change
Successfully updated the TMGC component c1057ef0-ab4e-459f-8413-0e5036c65b7c for
components of type logservice
Successfully imported configuration properties for the scope map[cluster:tml52clusterpune1
component:c1057ef0-ab4e-459f-8413-0e5036c65b7c zone:eastus] for components of type
logservice

```

For more information, see [JSON Configuration Files Format for Different Outputs](#).

JSON Configuration Files Format for Different Outputs

The following section contains example JSON output for different outputs.

For DATABASE output:

```
{
  "td_agent_output_channelType": "DATABASE",
  "td_agent_out_dbType": "<MYSQL or POSTGRESQL>",
  "td_agent_out_dbHost": "<network reachable database host>",
  "td_agent_out_dbPort": "<port for database>",
  "td_agent_out_dbSchema": "<name of the database>",
  "td_agent_out_dbUser": "<db_user>",
  "td_agent_out_dbPassword": "<db_password>"
}
```

For FORWARD output:

```
{
  "td_agent_container_output_channelType": "FORWARD",
  "td_agent_metric_output_channelType": "FORWARD",
  "td_agent_output_channelType": "FORWARD",
  "td_agent_verbose_output_channelType": "FORWARD",
  "td_agent_out_forward_servers": "fluentHost1:PORT,fluentHost2:PORT",
  "td_agent_out_metric_forward_servers": "fluentHost1:PORT, fluentHost2:PORT",
  "td_agent_out_container_forward_servers": "fluentHost1:PORT, fluentHost2:PORT",
  "td_agent_out_verbose_forward_servers": "fluentHost1:PORT, fluentHost2:PORT"
}
```



Note: Every type of log can be sent to the same FORWARD service.

For ELASTICSEARCH output:

```
{
  "td_agent_output_channelType": "DEFAULT,ELASTICSEARCH",
  "td_agent_container_output_channelType": "DEFAULT,ELASTICSEARCH",
  "td_agent_metric_output_channelType": "DEFAULT,ELASTICSEARCH",
  "td_agent_verbose_output_channelType": "DEFAULT,ELASTICSEARCH",
  "td_agent_out_elasticsearch_host": "<Elasticsearch Host IP>",
  "td_agent_out_elasticsearch_port": "9200",
  "td_agent_out_elasticsearch_index": "ml52_access",
  "td_agent_out_container_elasticsearch_host": "<Elasticsearch Host IP>",
  "td_agent_out_container_elasticsearch_port": "9200",
  "td_agent_out_container_elasticsearch_index": "tml52_container",
  "td_agent_out_metric_elasticsearch_host": "<Elasticsearch Host IP>",
  "td_agent_out_metric_elasticsearch_port": "9200",
  "td_agent_out_metric_elasticsearch_index": "ml52_metric",
}
```

```

    "td_agent_out_verbose_elasticsearch_host" : "<Elasticsearch Host IP>",
    "td_agent_out_verbose_elasticsearch_port" : "9200",
    "td_agent_out_verbose_elasticsearch_index" : "ml52_verbose"
  }

```

For KAFKA output:

```

{
  "td_agent_output_channelType": "DEFAULT,KAFKA",
  "td_agent_container_output_channelType": "DEFAULT,KAFKA",
  "td_agent_metric_output_channelType": "DEFAULT,KAFKA",
  "td_agent_verbose_output_channelType": "DEFAULT,KAFKA",
  "td_agent_out_kafka_brokers" : "<Kafka Host IP>:9092",
  "td_agent_out_kafka_topic" : "tml52_access_logs",
  "td_agent_out_metric_kafka_brokers" : "<Kafka Host IP>:9092",
  "td_agent_out_metric_kafka_topic" : "tml52_metric_logs",
  "td_agent_out_container_kafka_brokers" : "<Kafka Host IP>:9092",
  "td_agent_out_container_kafka_topic" : "tml52_container_logs",
  "td_agent_out_verbose_kafka_brokers" : "<Kafka Host IP>:9092",
  "td_agent_out_verbose_kafka_topic" : "tml52_verbose_logs"
}

```

For HTTP output:

```

{
  "td_agent_output_channelType": "DEFAULT,HTTP",
  "td_agent_container_output_channelType": "DEFAULT,HTTP",
  "td_agent_metric_output_channelType": "DEFAULT,HTTP",
  "td_agent_verbose_output_channelType": "DEFAULT,HTTP",
  "td_agent_out_http_URI": "<HTTP URI>",
  "td_agent_out_metric_http_URI": "<HTTP URI>",
  "td_agent_out_container_http_URI": "<HTTP URI>",
  "td_agent_out_verbose_http_URI": "<HTTP URI>"
}

```

For TCP output:

```

{
  "td_agent_output_channelType": "DEFAULT,TCP",
  "td_agent_container_output_channelType": "DEFAULT,TCP",
  "td_agent_metric_output_channelType": "DEFAULT,TCP",
  "td_agent_verbose_output_channelType": "DEFAULT,TCP",
  "td_agent_out_tcp_host": "<TCP HOST>",
}

```

```

"td_agent_out_tcp_port": "6000",
"td_agent_out_metric_tcp_host": "<TCP HOST>",
"td_agent_out_metric_tcp_port": "6000",
"td_agent_out_container_tcp_host": "<TCP HOST>",
"td_agent_out_container_tcp_port": "6000",
"td_agent_out_verbose_tcp_host": "<TCP HOST>",
"td_agent_out_verbose_tcp_port": "6000"
}

```

For SYSLOG output:

```

{
  "td_agent_output_channelType": "DEFAULT,SYSLOG",
  "td_agent_container_output_channelType": "DEFAULT,SYSLOG",
  "td_agent_metric_output_channelType": "DEFAULT,SYSLOG",
  "td_agent_verbose_output_channelType": "DEFAULT,SYSLOG",
  "td_agent_out_syslog_host": "<SYSLOG HOST>",
  "td_agent_out_syslog_port": "5142",
  "td_agent_out_syslog_tag": "ml5_accesslogs",
  "td_agent_out_syslog_protocol": "tcp",
  "td_agent_out_metric_syslog_host": "<SYSLOG HOST>",
  "td_agent_out_metric_syslog_port": "5142",
  "td_agent_out_metric_syslog_tag": "ml5_metriclogs",
  "td_agent_out_metric_syslog_protocol": "tcp",
  "td_agent_out_container_syslog_host": "<SYSLOG Host>",
  "td_agent_out_container_syslog_port": "5142",
  "td_agent_out_container_syslog_tag": "ml5_containerlogs",
  "td_agent_out_container_syslog_protocol": "tcp",
  "td_agent_out_verbose_syslog_host": "<SYSLOG Host>",
  "td_agent_out_verbose_syslog_port": "5142",
  "td_agent_out_verbose_syslog_tag": "ml5_verboselogs",
  "td_agent_out_verbose_syslog_protocol": "tcp"
}

```

JSON file with multiple output:

```

{
  "td_agent_output_channelType": "DEFAULT,ELASTICSEARCH,KAFKA",
  "td_agent_container_output_channelType": "DEFAULT,ELASTICSEARCH,KAFKA",
  "td_agent_metric_output_channelType": "DEFAULT,ELASTICSEARCH,KAFKA",
  "td_agent_verbose_output_channelType": "DEFAULT,ELASTICSEARCH,KAFKA",
  "td_agent_out_elasticsearch_host": "<Elasticsearch Host IP>",
  "td_agent_out_elasticsearch_port": "9200",
  "td_agent_out_elasticsearch_index": "ml52_access",

```

```

"td_agent_out_container_elasticsearch_host" : "<Elasticsearch Host IP>",
"td_agent_out_container_elasticsearch_port" : "9200",
"td_agent_out_container_elasticsearch_index" : "tml52_container",
"td_agent_out_metric_elasticsearch_host" : "<Elasticsearch Host IP>",
"td_agent_out_metric_elasticsearch_port" : "9200",
"td_agent_out_metric_elasticsearch_index" : "ml52_metric",
"td_agent_out_verbose_elasticsearch_host" : "<Elasticsearch Host IP>",
"td_agent_out_verbose_elasticsearch_port" : "9200",
"td_agent_out_verbose_elasticsearch_index" : "ml52_verbose",
"td_agent_out_kafka_brokers" : "<Kafka Host IP>:9092",
"td_agent_out_kafka_topic" : "tml52_access_logs",
"td_agent_out_container_kafka_brokers" : "<Kafka Host IP>:9092",
"td_agent_out_container_kafka_topic" : "tml52_container_logs",
"td_agent_out_metric_kafka_brokers" : "<Kafka Host IP>:9092",
"td_agent_out_metric_kafka_topic" : "tml52_metric_logs",
"td_agent_out_verbose_kafka_brokers" : "<Kafka Host IP>:9092",
"td_agent_out_verbose_kafka_topic" : "tml52_verbose_logs"
}

```

Local Edition Access Log Sync

The Access Log Sync process is deployed as a standalone running a Java process inside the **Log-service** container. Its primary job is to sync the generated access logs to the MOM environment via an HTTP call.

The access log sync is applicable only for tethered mode setup.

Prerequisites

- MOM host specified should be accessible via direct HTTPS connection from Boomi Cloud™ API Management - Local Edition cluster.
- The properties **apiKey** and **apiSecret** are required for MOM connection need to be passed through *tml_log_properties.json* as described in the section **Properties for access log sync to MOM (tethered mode)** in [Exporting Access Logs](#).

Process location (tml-log container)

/opt/mashery/access-log-sync/

/bin - Startup script for access-log-sync process, for example: startup_accesslog_sync.sh

/conf - Process configuration, for example: logsync.properties

/lib - Library jars location

/lib/tibco - TIBCO internal libraries

/lib/tpcl - External libraries

Process startup

The Access Log Sync process is a part of the **Log-service** container, and its lifecycle is managed by the log service.

Startup:

- Start up of the Log Sync process is managed by the parent script
/opt/mashery/containeragent/bin/startup_logservice.sh script.
- This script is called by containeragent itself on container setup and it starts access-log-sync after it starts td-agent process.
- Lifecycle options such as start/stop/restart/status of log service governs the same for both process (in the following order):
 - startup_td-agent.sh
 - startup_accesslog_sync.sh

Deployment Mode

- Tethered mode (MOM Sync is enabled): In this mode, log-sync will be running as a Java process, and it will perform the following:
 - Identify logs that need to be sent - access logs will continue getting written in default location, for example, /mnt/data/trafficmanager/access/access_worker<n>.log. Currently the access log rotates based on the configuration parameter out_mfileout.rb: log_max_size. The default value is 1MB.

1. After every write of `log_max_size`, the existing `access.log` file will be rotated to `access.log.1`, `access.log.2` and so on.
2. If the total size of the rotated log files exceeds the sync data size limit (default ~10 MB), it triggers the Log sync job.
3. After the Log sync job is triggered, it will move all rotated log files to the `/mnt/data/trafficmanager/queue/` folder. After moving, these files are compressed to a .zip file with the name, `log_epochFrom_log_epochTo_machineInstanceName.zip`.
4. Apart from the above size-based trigger, the Log sync job will continue to run on every configured sync time interval defined in `logsync.properties` (default: 15 min).



Note: In the scheduler sync, if access log files are not rotated (doesn't exceed `log_max_size` : 1 MB), the access log accumulated so far will still be synced with the name of `access.log.0` and moved to archive directory.

5. The total size of the compressed log files should not exceed 10 MB (configurable). After this limit, the .zip file will not include any further log files for sync call.
- Data enrichment on log-sync.
 1. Read logs from queue folder.
 2. Send ONLY access logs from Traffic Manager containers - No need for cleanup and de-duplication.
 3. Enrichment of `data_source_id` and `capacity_center_id/cluster_id` - log-sync process will call MOM endpoint using credentials and fetch MOM metadata to enrich logs.
 - Archive already sent data.
 1. Once sent, move logs from `/mnt/data/trafficmanager/queue/` to `/mnt/data/trafficmanager/archive/timestamp/`
 - Maintain audit of sync done.
 1. Write all the metrics to a file in `/mnt/data/tmgc-log/{LOG_HOST_`

NAME}/accesslogconsumer/syncaudit/syncaudit.log in the form of key-value pair.

- Status of the sync process (running, stopped, etc.)
- Untethered mode (MOM Sync is disabled): In this mode, the access-log-sync though is configured but will not run. A check is put in the startup script to check if MOM Credentials are configured in /opt/mashery/access-log-sync/conf/logsync.properties. If not, it will not start.

Sync Interval change by client

You can change the sync interval in minutes [range 1-59] in the scope of a cluster/zone using CLI. For example:

```
Syntax: clustermanager set lsi --componentId <log_service_component_id> --componentType logservice --syncInterval 12
```

eg :

```
root@cm-deploy-0-777dd4ff67-vlfjz builder]# clustermanager set lsi --componentId 9d7e4de0-c11e-42b4-86b8-4295ff4286df --componentType logservice --syncInterval 12
```

```
Calling POST url: http://xxx.xx.x.x:9080/log/loggers/access/sync/configuration?syncinterval=12
Successfully updated log sync interval 12 for log component 9d7e4de0-c11e-42b4-86b8-4295ff4286df
[root@cm-deploy-0-777dd4ff67-vlfjz builder]#
```

This new sync Interval will be replaced in /opt/mashery/access-log-sync/conf/logsync.properties and will be readily picked up by LogSyncJob to start syncing according to this interval.

Logging of Access Log Sync

- Appender and logger is added in container agent logback.xml using Syslogappender.
- Process logs are stored in the folder, /mnt/data/tmgc-log/{LOG_HOST_NAME}/accesslogconsumer/consumer.log (directory name will be governed by metadata defined in logback.xml).
- Audit logs comprising of log sync result will be written to folder, /mnt/data/tmgc-log/{LOG_HOST_NAME}/accesslogconsumer/syncaudit/syncaudit.log

Payload Logging

- [What is payload logging](#)
- [What is captured in payloads](#)
- [Planning for Payload logging](#)
- [How to enable payload logging](#)
- [How do direct payloads to external destinations](#)
- [Constraints of payload logging](#)
 - [Estimating disk/volume size](#)
- [Tuning & Debugging](#)
 - [Tuning Parameters](#)
 - [Debugging and Troubleshooting](#)
 - [Where are payloads captured and logged](#)
 - [Matching counts of payloads and traffic](#)
- [Changing count batch sizes for logging](#)

- **What is payload logging**

Payload logging captures body content of requests and responses if any along with some metadata. This data is stored on log service pods/containers and users can also direct payload to external destinations. This feature is not turned on by default. It has to be enabled by user after a Local Edition cluster is started.

- **What is captured in payloads**

For every request the following are captured (where available)

1. Request metadata
2. Inbound request body content
3. Target request body content
4. Target response body content
5. Outbound response body content.

i **Note:** Please see the [Estimating disk/volume size](#) section for demands on storage.

The following table shows different attributes, captured as part of each payload

Payload type/Stage	Attribute	Notes
Request metadata	requestUUID	A unique identifier for each request.
	requestURI	The Endpoint that was invoked.
	cacheHit	A boolean which signifies if request was served from the cache.
	statusCode	Outbound status code.
	statusMessage	Outbound status message.
	endpointKey	
	spKey	
	key	The MAPI key used to make the call.
	payload_event	A tag used to identify the event. <i>request_metadata</i> in this case.
Inbound request	requestUUID	
	payload_event	A tag used to identify the event. <i>inbound_request</i> in this case.

Payload type/Stage	Attribute	Notes
	bodyContent	One of 1. Payload from request. 2. The text NO BODY CONTENT in case the request does not have a body content. 3. The text ERROR READING BODY CONTENT in case of any error reading the body content.
Target request	requestUUID	
	payload_event	A tag used to identify the event. <i>target_request</i> in this case.
	bodyContent	One of 1. Payload for request to target. 2. The text NO BODY CONTENT in case the request does not have a body content. 3. The text ERROR READING BODY CONTENT in case of any error reading the body content.
Target response	requestUUID	

Payload type/Stage	Attribute	Notes
	payload_event	A tag used to identify the event. <i>target_response</i> in this case
	bodyContent	One of 1. Payload in response from target. 2. The text NO BODY CONTENT in case the request does not have a body content. 3. The text ERROR READING BODY CONTENT in case of any error reading the body content.
	targetStatusCode	status code of request from backend
	targetStatusMessage	status message of request from backend
Outbound response	requestUUID	
	payload_event	A tag used to identify the event. <i>outbound_response</i> in this case
	bodyContent	One of 1. Payload in response from target. 2. The text NO BODY CONTENT in case the request does not have a body content. 3. The text ERROR READING BODY CONTENT in case of any error reading the body content.

- Planning for payload logging

1. At high QPS with payload logging enabled; the demand for disk space will go up. To ensure that disk utilization and limits do not go out of bounds, by default payloads are not logged on the log pods.
2. The disk cleanup utility on log service will empty payloads when utilization reaches 70 percent of disk size.
3. Therefore before creating the API Management - Local Edition cluster.
 - a. Increase the storage requirements as per QPS expected.
 - b. Decide and define the config for directing payloads to external destinations.

- **How to enable payload logging**

User can turn on payload logging by using the clustermanager command line.

1. login into the CM container.
2. run the command `clustermanager set payloadlogging --enable=<true|false>`

- **How do direct payloads to external destinations**

Payloads can be sent to all supported destination types except DATABASE.

Payloads can be sent to the following destinations: DEFAULT, ELASTICSEARCH, FORWARD, KAFKA, HTTP, SYSLOG and TCP. Changing to anything other than DEFAULT requires additional properties.

Destination type	Name in JSON	Description	Notes	Example
	td_agent_payload_logging_output_channelType	Comma separated string of types of channels	One of ELASTICSEARCH, FORWARD, HTTP, KAFKA, SYSLOG, TCP	"td_agent_payload_logging_output_channelType": "DEFAULT,ELASTICSEARCH"
ELASTICSEARCH	td_agent_out_payload_logging_output_channelType	Host	The hostname	"td_agent_out_payload_logging_output_channelType": "ELASTICSEARCH"

Destination type	Name in JSON	Description	Notes	Example
C search	out_payload_logging_elasticsearch_host	name or IP address of Elasticsearch server or ELB	should be reachable from within the log service container.	elasticsearch_host" : "a06ebe8a15a3011e9bf380608b2b6e73-531419666.us-east-1.elb.amazonaws.com"
	td_agent_out_payload_logging_elasticsearch_port			"td_agent_out_payload_logging_elasticsearch_port" : "9200"
	td_agent_out_payload_logging_elasticsearch_index		Defaults to incoming log event tag.	"td_agent_out_payload_logging_elasticsearch_index" : "ml5_payload_logging_logs"
	td_agent_out_payload_logging_elasticsearch_protocol	The protocols supported are HTTP and HTTPS.	This is optional. The default protocol is HTTP.	"td_agent_out_payload_logging_elasticsearch_protocol": "https"
	td_agent_out_payload_logging_elasticsearch_ssl_version	SSL version configured on the	Use this if td_agent_out_payload_	"td_agent_out_payload_logging_elasticsearch_ssl_version": "TLSv1_2"

Destination type	Name in JSON	Description	Notes	Example
	elasticsearch_ssl_version	destination elastic search instance.	logging_elasticsearch_protocol is HTTPS. Supported versions are TLSv1_2, TLSv1, TLSv1_1,SSLv23. Default version is TLSv1_2.	
	td_agent_out_payload_logging_elasticsearch_cafile	The Name of CA file to be used.	Use this if td_agent_out_payload_logging_elasticsearch_protocol isHTTPS . The value should be the name of the file.	"td_agent_out_payload_logging_elasticsearch_cafile":" cafile-a06ebe8a15a3011e9bf380608b2-531419666.us-east-1.elb.amazonaws.com .crt"
	td_agent_out_payload_logging_elasticsearch_user	User name to authenticate.	This is optional depending on the elastic search security settings.	"td_agent_out_payload_logging_elasticsearch_user":"elasticuser"
	td_agent_out_	Credential	This is optional	"td_agent_out_payload_logging_elasticsearch_

Destination type	Name in JSON	Description	Notes	Example
	payload_logging_elasticsearch_password		depending on the elastic search security settings.	password":"password123"
FORWARD	td_agent_out_payload_logging_forward_servers	Hostname or IP address and port for available FluentD servers Expected value is [{HOST}: {PORT}, {HOST1}: {PORT}...] IF NO PORT GIVEN, DEFAULT PORT 24224 will be considered.	The hostname should be reachable from the log service container.	"td_agent_out_payload_logging_forward_servers" : "192.168.1.11:24221,192.168.2.11:24225" OR "td_agent_out_payload_logging_forward_servers" : "fluentHost1:24221,fluentHost2:24225" Without PORT example "td_agent_out_payload_logging_forward_servers" : "fluentHost1:,fluentHost2:" It will connect with default port 24224

Destination type	Name in JSON	Description	Notes	Example
		NOTE: Data will be pushed to any of the available server. if not then LogService service look for another available one.		
HTTP	td_agent_out_payload_logging_http_URI	HTTP URI to which logs are to be sent.	The HTTP URI should be accessible from the log container.	"td_agent_out_payload_logging_http_URI": "http://a06ebe8a15a3011e9bf380608b2b6e73-531419666.us-east-1.elb.amazonaws.com/ml5/payload_logginglogs"
KAFKA	td_agent_out_payload_logging_kafka_brokers	Comma separated list of kafkabroker_host:port.	The Kafka brokers host should be reachable from the log service and broker should be running.	"td_agent_out_payload_logging_kafka_brokers": "broker1:9092,broker2:9092"

Destination type	Name in JSON	Description	Notes	Example
	td_agent_out_payload_logging_kafka_topic	Topic on which messages have to be sent.		"td_agent_out_payload_logging_kafka_topic": "ml5_payload_logging_logs"
SYSLOG	td_agent_out_payload_logging_syslog_host	Host name of IP address where syslog is running.	Host should be reachable from log service.	"td_agent_out_payload_logging_syslog_host": "a06ebe8a15a3011e9bf380608b2b6e73-531419666.us-east-1.elb.amazonaws.com"
	td_agent_out_payload_logging_syslog_port	Port number of syslog program.		"td_agent_out_payload_logging_syslog_port": "5142"
	td_agent_out_payload_logging_syslog_tag	Tag with which the log is to be uniquely identified.	Defaults to incoming log event tag.	"td_agent_out_payload_logging_syslog_tag": "ml5_payload_logging_logs"
	td_agent_out_payload_logging_protocol	Protocol on which syslog is listening	Defaults to tcp.	"td_agent_out_payload_logging_syslog_protocol": "tcp"

Destination type	Name in JSON	Description	Notes	Example
	syslog_protocol	for incoming requests.		
TCP	td_agent_out_payload_logging_tcp_host	Host name where a TCP socket is listening.	Host should be reachable from log service.	"td_agent_out_payload_logging_tcp_host": "a06ebe8a15a3011e9bf380608b2b6e73-531419666.us-east-1.elb.amazonaws.com"
	td_agent_out_payload_logging_tcp_port	Port on which the TCP socket is listening.		"td_agent_out_payload_logging_tcp_port": "6000"

- **Constraints of payload logging**

1. Payload size of max 1MB are logged. Extra bytes are discarded.
2. Payload logging is mutually exclusive to verbose logging due to high disk utilization.
3. Enabling payload logging WILL impact QPS performance.
4. Payloads of the following stages in a traffic call are logged.
 - a. Request
 - b. Target request
 - c. Target response
 - d. Response
 - e. Request metadata
 - **Estimating disk/volume size**

The max payload logged is 1 mb. Assuming all 4 stages log 1 mb of payload.

Number of stages	4	
data per stage	1mb	
Total per request	number of stages * data per stage	4*1 = 4mb
Estimated QPS	2k per second	
Storage demand for payload logging per second	2000*4mb	2000*4MB = 8000MB or 8GB per second

This a broad estimation; typical request body content won't be as large as 1 MB except in certain cases like attachments but response body contents can exceed 1MB. Assuming a range of 500KB to 1MB of body content; a nearness estimate will be 2GB to 4GB of storage demand per second for 2k QPS.

Tuning and Debugging

- **Tuning parameters**

For every traffic call, the traffic manager dispatches 5 events on a separate thread. So if the jetty pool threads are modified; it is recommended to set the number of payload threads to the same value. The following table shows the properties that correlate between jetty and payload logging.

Payload Logging	Jetty/Traffic	Notes
logger_maxthreads	jetty_pool_maxthreads	Both the values must be same
logger_acceptqueuesize	jetty_acceptqueuesize	Both the values must be same

- **Debugging and trouble shooting**

By default the payloads are not written to disk as the velocity and volume of writes will cause a degradation in the performance of the log pod. Instead, it is recommended to configure an external destination for receiving payloads. ([Refer](#)

the table). But to test if payloads are being received properly or to verify the contents of payloads in test environments; it may be necessary to write to disk.

- **Enabling payloads to be written to disk inside log pod**

```
clustermanager set config --componentType logservice --prop payload_logging_debug --value true
```

Disabling payloads to be written to disk

```
clustermanager set config --componentType logservice --prop payload_logging_debug --value false
```



Note: It is recommended to turn off payloads being written to disk to ensure that the performance of the log pods' don't degrade.

In tethered mode, if the AreaConfig does not have proper values set for the proxy_max_ci_threads_per_customer_per_host and sometimes the tethered mode may raise a LoggerImpl :: ciPerAreaThrottleService.tryAcquire(config) = false, LIMIT breached for logger factory error in its logs, then follow the steps below to resolve this error:

1. Login to tml-sql POD/Container.
2. Run the mysql -uroot -pchangeme command to connect to mysql shell.
3. Run the USE masherysolar command to change the database.
4. Run the update command:

```
UPDATE area_config AS AC SET AC.eav=REPLACE(AC.eav, '<proxy_max_ci_threads_per_customer_per_host>2</proxy_max_ci_threads_per_customer_per_host>', '<proxy_max_ci_threads_per_customer_per_host>512</proxy_max_ci_threads_per_customer_per_host>') where area_id = <customer_area_id>;
```

Once the cache update schedule runs, the changes are available in the tethered mode to use.

- **Where are payloads captured and logged**

The payloads are captured on log service pods/containers at the location /mnt/data/payloads.

A directory for each request is created by using the value of requestUUID as the name of the directory.

Under each requestUUID directory; 5 files are available; one each for each of the events. The event name from payload_event attribute is the file name. In the case of the request being served from the cache; the target request and response file would not be present. The contents of each event are stored as json data.

- **Matching counts of payloads and traffic**

The payload logs batch counts can be logged by enabling the logger PAYLOAD_COUNTS_LOGGER. By default, this logger is turned off. to turn it on using the cluster manager.

```
clustermanager set loglevel --loggerName PAYLOAD_COUNTS_LOGGER --logLevel INFO --
componentType trafficmanager
```

The logger logs separate batch counts for each of the different payloads (link to a table in What is captured in payloads). It also logs separately the batch counts for successful writes, fails, and misses.

Example log lines

```
[2021-02-03T16:21:37+00:00] INFO [CILoggerPool-LoggerThread-2] PAYLOAD_
COUNTS_LOGGER Payload event type TARGETREQUESTS: flush log count
100000
```

```
[2021-02-03T16:21:37+00:00] INFO [CILoggerPool-LoggerThread-4] PAYLOAD_
COUNTS_LOGGER Payload event type REQUESTS: flush log count 100000
```

```
[2021-02-03T16:21:37+00:00] INFO [CILoggerPool-LoggerThread-4] PAYLOAD_
COUNTS_LOGGER Payload event type REQUESTMETADATA: flush log count
100000
```

```
[2021-02-03T16:21:37+00:00] INFO [CILoggerPool-LoggerThread-3] PAYLOAD_
COUNTS_LOGGER Payload event type TARGETRESPONSES: flush log count
100000
```

```
[2021-02-03T16:21:37+00:00] INFO [CILoggerPool-LoggerThread-1] PAYLOAD_
COUNTS_LOGGER Payload event type RESPONSES: flush log count 100000
```

Summing up the batch counts of all traffic manager for an event - for example, REQUESTMETADATA; we can match that total to the number of files on the log pod matching the file name. request_metadata.json

Changing Count Batch Sizes for Logging

Name	Default	Notes
payload_ success_ count_ flush_limit	100000	The count of successful submission by the traffic manager of payload for logging for each type of payload.
payload_ miss_ count_ flush_limit	1	The count of misses of submission by the traffic manager of payload for logging for each type of payload. Misses happen when a failure is previously encountered. A batch size of 1 will give an accurate measure of how many requests were missed.
payload_ fail_count_ flush_limit	10	The count of failure of logging payloads can happen because the connection between the traffic manager and LFA is terminated for some reason; when the connection is closed, the traffic manager tries to reconnect.



Note: These counts are only for submission by the traffic manager; these do not reflect the actual capture of the payload by the destination.

HTTPS Configuration

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 content area is titled "Project upload_ssl_server_truststore_for_traffic_manager" and states "This build requires parameters:". Below this, 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 input field and the text "SSL truststore password". A "Build" button is located below these fields. At the bottom left, there is a "Build History" section showing a search bar with "find", a list of builds with the first one being "#1" from "Feb 14, 2020 11:40 PM", and 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:

```
"tml_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
```

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

Security

Performance Acceleration

Portal Access Groups

Endpoints

Error Sets

Home / API Definitions / Hw_sample_service

Hw_sample_service

Save Cancel

API Service Key

uyhejgmusp7kwdjz6xf5qbcm

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

Modify the description of the API. You can enter purpose, limitations, or other important information about the API.

Version

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

Enter the data for cross-domain policy for API that can be accessed by using Flash.

Robots Policy

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			
Security	Endpoints			
Performance Acceleration	my search terms Search + New			
Portal Access Groups				
Endpoints	Name	Updated	URL	Actions
Error Sets	hw_sample_endpoint	Nov 17 2021 20:17	http://api.example.com/hello	

- 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

Identity Store Name:

one_way_profile_001

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:

1. 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=sra663fmnshjazx5sv2mgmtmw'
{"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=sra663fmnshjazx5sv2mgmtmw' -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.

Mutual HTTPS Client Configuration

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	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	identity_011	5488
cb4d2bc2-f863-440b-0012-1d11369404d8	identity_012	5488
cb4d2bc2-f863-440b-0013-1d11369404d8	identity_013	5488
cb4d2bc2-f863-440b-0014-1d11369404d8	identity_014	5488
cb4d2bc2-f863-440b-0015-1d11369404d8	identity_015	5488
cb4d2bc2-f863-440b-0016-1d11369404d8	identity_016	5488
cb4d2bc2-f863-440b-0017-1d11369404d8	identity_017	5488
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.

```
kubectcl 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	identity_011		5488
cb4d2bc2-f863-440b-0012-1d11369404d8	identity_012		5488
cb4d2bc2-f863-440b-0013-1d11369404d8	identity_013		5488
cb4d2bc2-f863-440b-0014-1d11369404d8	identity_014		5488
cb4d2bc2-f863-440b-0015-1d11369404d8	identity_015		5488
cb4d2bc2-f863-440b-0016-1d11369404d8	identity_016		5488

cb4d2bc2-f863-440b-0017-1d11369404d8	identity_017	5488
cb4d2bc2-f863-440b-0018-1d11369404d8	identity_018	5488
cb4d2bc2-f863-440b-0019-1d11369404d8	identity_019	5488
cb4d2bc2-f863-440b-0020-1d11369404d8	identity_020	5488

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

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

Creating Mutual Authentication

Ensure you have your setup according to one of the following modes (configured when setting up Configuration data during initial deployment):

- [Tethered mode](#)
- [Untethered mode](#)

Tethered Mode

This mode of configuration in Local Edition fetches the certificates and identities, as well as mutual authentication profiles from the MOM.

Follow these steps to use tethered mode:

1. Create an [identity](#).
2. Create a [certificate](#).
3. Up-sync with MOM to upload the identity and certificate.
4. The HTTPS Client Profile can now be associated with the identity and/or certificate.

i Note: Creating an HTTPS Client Profile in tethered mode in Local Edition 5.x is the same as in Local Edition 4.x. See [Configuring and Using the HTTPS Client Feature with Mutual Authentication](#).

5. The next cloud sync with Local Edition through the MOM will down-sync the HTTPS Client Profile to Local Edition.

The Identity and/or Certificate data can be accessed and modified using the [Cluster Manager CLI commands](#).

Untethered Mode

This mode configuration mode, all API configurations and their policies are managed through the V3 API layer.

In order to access the V3 APIs, first, create an access token using your specific credentials, such as: 4g4d3zaxnbvp9ee7kdnum8as:JTJCrVeW7d as shown in the examples below:

1. Begin by creating an access token.

```
curl -v -u 4g4d3zaxnbvp9ee7kdnum8as:JTJCrVeW7d -H 'Host: api.tml.mashspud.com'
'http://a981e26fd5b1f11e98d7b06eb89ed329-73173243.us-east-
1.elb.amazonaws.com:7080/v3/token' -d "grant_
type=password&username=localdevadmin&password=ch4ng3m3&scope=84e9d6f1-920d-
4b6c-9289-5432c6fd39b7"
```

2. Create an identity:
 - See [Create Identity](#).
 - See [List Identities](#). (This ID is referred to as "YOUR_IDENTITY_UUID" in the

code example below.)

3. Create a certificate:

- See [Create Certificate](#).
- See [List Certificates](#). (This ID is referred to as "YOUR_TRUST_CERT_UUID" in the code example below.)

4. Create an HTTP Client Profile.

Use the access token from the first call in the request header, e.g.

"xhah3bv2acgnxjmtun7fwhh8" and associate the identity and the certificate to the https client profile.

```
curl -H 'Authorization: Bearer xhah3bv2acgnxjmtun7fwhh8' -H 'Content-Type: application/json' -H 'Host: api.tml.mashspud.com'
'http://a981e26fd5b1f11e98d7b06eb89ed329-73173243.us-east-1.elb.amazonaws.com:7080/v3/rest/httpsClientProfiles' -d '{"name":"TestHttpClientProfile", "description":"Test Description", "verifyHostname": false, "httpsClientTrusts":[{"id":"YOUR_TRUST_CERT_UUID"}], "identity":{"id":"YOUR_IDENTITY_UUID"}'}
```

Response:

```
{ "created": "2019-04-18T20:54:22.745+0000", "updated": "2019-04-18T20:54:22.745+0000", "sni": false, "verifyHostname": false, "httpsClientTrusts": [{"created": "2019-04-18T20:54:22.745+0000", "updated": "2019-04-18T20:54:22.745+0000", "id": "5cdf6dba-322f-410a-b4e7-75083e87e752"}], "identity": {"serialNumber": "5c9a9404", "created": "2019-04-03T23:53:59.000+0000", "updated": "2019-04-03T23:53:59.000+0000", "expiration": "2020-03-27T04:05:09.000+0000", "fingerprint": "DB:13:C6:CF:79:79:C4:6F:29:AE:0B:7E:3D:89:A2:5F", "name": "TestHttpClientProfile", "id": "0f825c1c-b01e-41b2-80f1-6d6bdc4b39d1"}, "description": "Test Description", "name": "TestHttpClientProfileFromCall", "id": "2415970a-fc85-4029-acab-9a349334838f" }
```

5. The HTTPS Client Profile can now be associated with the identity and/or certificate.

OAuth

- [Configuring OAuth 2.0 API Access](#)
- [Making OAuth 2.0 Calls](#)
- [5.1 OAuth API Compatibility with 4.x](#)
- [OAuth Token Migration](#)

Configuring OAuth 2.0 API Access

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": "3bvu8u3p8l",
  "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 "cm9vdDpjaGFuZ2VtZQ==" in the curl header -H 'Authorization: Basic cm9vdDpjaGFuZ2VtZQ==' 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":"m9f5kyzfjrrsz6d5sfkbuw3","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":"m9f5kyzfjrrsz6d5sfkbuw3","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"
}
```

```
"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":"m9f5kyzfjrrsz6d5sfkbuw3","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":"unrct8swseny8h3b3kw39wab","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":"m9f5kyzfjrrsz6d5sfkbuw3","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":""}},"error":null}
```

Create access token with authorization code

```
curl -v -d '{"id":1,"method":"oauth2.createAccessToken","params":{"service_
key":"m9f5kyzfjrrsz6d5sfkbuw3","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":"zgwyg64s2ejc6catffvg5kxb","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":"m9f5kyzfjrrsz6d5sfkbuw3","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":"m9f5kyzfjrrsz6d5sfkbuw3","access_token": "rjzb2k26cg7tufe2j8cs7rm"}}' -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":"m9f5kyzfjrrsz6d5sfkbuw3","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":"TestOAuthRegisterCallBack"}}
```

Fetch user application

```
curl -v -d '{"id":1, "method":"oauth2.fetchUserApplications", "params":{"service_
key":"m9f5kyzfjrrsz6d5sfkbuw3", "user_context":"testUser1455730448397"}' -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":"TestOAuthRegisterCallBack","access_tokens":
["rjzb2k26cgv7tufe2j8cs7rm","unrct8wsweny8h3b3kw39wab"],"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":"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: 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":"m9f5kyzfjrrsz6d5sfkbuw3", "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":"m9f5kyzfjrrsz6d5sfkbuw3","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":null}}
```

Sample direct calls with Basic Authentication

Create access token

```
curl -v -d '{"id":1,"method":"oauth2.createAccessToken","params":{"service_
key":"m9f5kyzfjrrsz6d5sfkbuw3","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":"m9f5kyzfjrrsz6d5sfkbuw3","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":""}},"error":null}

```

Create access token with authorization code

```

curl -v -d '{"id":1,"method":"oauth2.createAccessToken","params":{"service_
key":"m9f5kyzfjrrsz6d5sfkbuw3","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":null}}

```

Update access token

```

curl -v -d '{"id":1,"method":"oauth2.updateAccessToken","params":{"service_key":"m9f5kyzfjrrsz6d5sfkbuw3","client":{"client_id":"yq7chp6ufkeea587gp6wqa6x","client_secret":"Az4Xa"},"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":"s4b7qs24qfns2m3ecjmuvgh4","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": "s4b7qs24qfns2m3ecjmuvgh4"}}'
'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":"m9f5kyzfjrrsz6d5sfkbuw3", "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":"s4b7qs24qfns2m3ecjmuvg4" }, "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":"m9f5kyzfjrrsz6d5sfkbuw3", "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":"m9f5kyzfjrrsz6d5sfkbuw3","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":null}}
```

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":"m9f5kyzfjrrsz6d5sfkbuw3","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":null}}
```

Create authorization code

```
curl -v -d '{"id":1,"method":"oauth2.createAuthorizationCode","params":{"service_key":"m9f5kyzfjrrsz6d5sfkbuw3","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":"m9f5kyzfjrrsz6d5sfkbuw3","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":"s4b7qs24qfns2m3ecjmuvg4","expires_in":36000,"refresh_token":"hekntywdyp2hg7yvfba5g","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":"m9f5kyzfjrrsz6d5sfkbuw3","client":{"client_id":"yq7chp6ufkeea587gp6wqa6x","client_secret":"Azn4Xa"},"access_token":"s4b7qs24qfns2m3ecjmuvg4","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":"m9f5kyzfjrrsz6d5sfkbuw3","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":"m9f5kyzfjrrsz6d5sfkbuw3","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":"m9f5kyzfjrrsz6d5sfkbuw3", "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": "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/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":"m9f5kyzfjrrsz6d5sfkbuw3", "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":"m9f5kyzfjrrsz6d5sfkbuw3","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/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":null}}
```

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":"m9f5kyzfjrrsz6d5sfkbuw3","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":"unrct8wsweny8h3b3kw39wab","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":"m9f5kyzfjrrsz6d5sfkbuw3","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}
```

Create access token with authorization code

```
curl -v -d '{"id":1,"method":"oauth2.createAuthorizationCode","params":{"service_
key":"m9f5kyzfjrrsz6d5sfkbuw3","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":"m9f5kyzfjrrsz6d5sfkbuw3","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":"m9f5kyzfjrrsz6d5sfkbuw3","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":"m9f5kyzfjrrsz6d5sfkbuw3", "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","unrct8wsweny8h3b3kw39wab"],"client_
id":"yq7chp6ufkeea587gp6wqa6x"}]}
```

Revoke access token

```
curl -v -d '{"id":1, "method":"oauth2.revokeAccessToken", "params":{"service_
key":"m9f5kyzfjrrsz6d5sfkbuw3", "client": {"client_id":"yq7chp6ufkeea587gp6wqa6x", "client_
secret":"Az4Xa"}, "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":"m9f5kyzfjrrsz6d5sfkbuw3","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":"m9f5kyzfjrrsz6d5sfkbuw3","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":36000,"refresh_token":"7db67za3xvdg99nc7gnctkqw","scope":null,"user_context":"testUser1455730448397","uri":null,"extended":null,"state":null}}
```

Local Edition 5.x OAuth API Compatibility with Local Edition 4.x

The OAuth feature in Local Edition 5.x has two versions of API:

- /v2/json-rpc
- /v3/json-rpc

V2 is backward-compatible with the Local Edition 4.x OAuth API, whereas V3 is the recommended version of the API which is compatible with Boomi Cloud API Management. There is no difference in supported methods and input parameters, except for a few differences in the responses mentioned below.

Sample Calls to Local Edition 5.x OAuth API

For Local Edition 4.x response with /v2/json-rpc

```
curl -v -d '{ "method": "oauth2.createAccessToken", "params": { "service_key": "m9f5kyzfjrrsz6d5sfkbuw3", "client": { "client_id": "4c7fmwh59w3ufn7pxt92qwx9", "client_
```

```
secret":"UhgX9W"},"token_data":{"grant_type":"client_credentials","scope":"scope"},"uri":{"redirect_uri":"https://yahoo.com/cb"},"user_context":"testuser"},"id":1001,"jsonrpc":"2.0"}'
'https://x.x.99.100:8083/v2/json-rpc' -u root:changeme -k
```

For Local Edition 5.x response with /v3/json-rpc

```
curl -v -d '{"method":"oauth2.createAccessToken","params":{"service_key":"m9f5kyzfjrrsz6d5sfkbuw3","client":{"client_id":"4c7fmwh59w3ufn7pxt92qwx9","client_secret":"UhgX9W"},"token_data":{"grant_type":"client_credentials","scope":"scope"},"uri":{"redirect_uri":"https://yahoo.com/cb"},"user_context":"testuser"},"id":1001,"jsonrpc":"2.0"}'
'https://x.x.99.100:8083/v3/json-rpc' -u root:changeme -k
```

Differences between V2 and V3 Response

The V2 and V3 responses in OAuth API calls have the following differences:

- For some fields that were not returned in the response of the call to `/v3/json-rpc`, for example `grant_type`, `user_context`, or `scope`, these fields are added to the response in the `/v2/json-rpc` call.
- For the field `refresh_token_expires` in Local Edition 4.x OAuth API response, it is removed from the call for both `/v2/json-rpc` and `/v3/json-rpc` calls.
- All error format `error.error.error` in a call to `/v3/json-rpc` is converted to `error.data.error` in `/v2/json-rpc`. For example:

Mashery Local 5x Response	Mashery Local 4x Response
<pre>{ "jsonrpc": "2.0", "id": 110005, "error": { "message": "Invalid refresh token", "error": { "error": "invalid_grant", "error_description": "Invalid refresh token", "error_response": null }, "code": -2001 } }</pre>	<pre>{ "jsonrpc": "2.0", "id": 110005, "error": { "message": "Invalid refresh token", "data": { "error_description": "Invalid refresh token", "error": "invalid_grant" }, "code": -2001 } }</pre>

- If the response in `/v2/json-rpc` is different from the response from a Local Edition 4.x OAuth API call, it's probably due to a bug in Local Edition 4.x.

OAuth Token Migration

The OAuth Token Migration utility is a stand-alone program that copies active OAuth tokens between a Local Edition 4.x (or below) environment and Local Edition 5.x installation. The utility can work off different combinations of source and destination via:

- MySQL
- File (export records to file and import of records from FILE)
- Cassandra

i Note: Where possible, the recommended source is MySQL and recommended destination is Cassandra.

Prerequisites

The container where the utility is running should be able to connect to source and destination data repository (MySQL/Cassandra) for fetching and writing of records.

In case the connectivity is not possible, the utility can work on intermediate FILE based migration where it should be like MySQL=> FILE => CASSANDRA.

While writing to FILE, the File output directory (mentioned in config file) should be large enough to hold records from source.

Structure of Token Migration Utility

The Token Migration utility is a self-contained application packaged with supporting libraries, configuration files, and startup script. When the utility tar is extracted, it consists of the following folders:

- **/bin:** Consists of startup scripts to execute the migration utility.
- **/lib:** Consists of all supporting libraries this utility needs.
- **/conf:** Consists of configuration files which are used to define execution parameters for source and destination. Also it consists of performance-related configurations for utility and logback for the application.

After you run the migration first time, it additionally creates two directories:

- **/log:** Consists of application logs.

- **/data:** Consists of checkpoint file (described in [Running the OAuth Migration Token Utility](#).)

Deploying the OAuth Token Migration Utility

Downloading the Utility

The OAuth Token Migration utility is part of the Installer.

Setting up the Utility

Create the `/opt/mashery/migration/` directory where the migration utility will be deployed, using the following command:

```
$ mkdir /opt/mashery/migration/
```

Uploading and Installing the Utility

Note: The following steps assume that the Local Edition 5.x cluster has been created and configured.

Platform	Steps
AWS Kubernetes Cluster	1. Log in to the terminal from where you can access the pods. 2. Once the cluster is up, install the migration utility: a. Identify the Cluster Manager container name:

Identify Mashery Local CM Container

```
$ kubectl get pods
```

NAME	READY	STATUS	RESTARTS	AGE
cache-deploy-0-9f66654b8-775vs	1/1	Running	0	7m
cass-set-0-0	1/1	Running	0	8m
cm-deploy-0-6fcf4c44d-fv8vf	1/1	Running	0	7m
log-set-0-0	1/1	Running	0	7m
mysql-set-0-0	1/1	Running	0	7m
tm-deploy-0-66c55846b8-4nq4r	1/1	Running	0	7m

Platform	Steps
	b. Copy the migration utility tar to Cluster Manager container: <pre>kubectrl cp tml-token-migration-v1.0.0.tar.gz cm-deploy-0-6fcf4c44d-fv8vf:/opt/mashery/migration/</pre> c. Extract the tar and write/modify suitable configuration properties in tml-token-migration-v1.0.0/conf/config.properties: <pre>\$ cd /op/mashery/migration \$ tar -xvzf tml-token-migration-v1.0.0.tar.gz</pre> d. Edit config.properties: <pre>\$ cd tml-token-migration-v1.0.0/conf \$ vi config.properties</pre> Note: See the section OAuth Token Migration Configuration Properties for details of required configurations. e. Copy the exported data.tar archive to Cluster Manager container in order for this to be migrated to the destination (Cassandra): <pre>\$ kubectrl cp tokens.tar cm-deploy-0-6fcf4c44d-fv8vf:/opt/mashery/migration/ \$ tar -xvzf tokens.tar</pre>
AWS Docker Swarm Cluster	Once the cluster is up, install the migration utility: 1. Identify the Cluster Manager container name:

Platform

Steps

Identify Mashery Local CM Container						
<pre>\$ eval "\$(docker-machine env node-1)" \$ docker ps</pre>						
CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	
e8981a04607b	tml-tm:v5.1.0.86	"/usr/local/bin/tm-e..."	21 hours ago	Up 21 hours	22/tcp, 80/tcp, 443/tcp,	
dfeee8e46a12	tml-sql:v5.1.0.86	"/usr/local/bin/sql-..."	21 hours ago	Up 21 hours	3306/tcp	
b77fad139de0	tml-cm:v5.1.0.86	"/usr/local/bin/cm-e..."	21 hours ago	Up 21 hours		

Note: If the Cluster Manager container is running on a different node, repeat the previous step for every node (Master, etc.).

- Repeat the previous step, with this change in the command (kubectl should be replaced with docker), for example:

```
docker cp tml-token-migration-v1.0.0.tar.gz
b77fad139de0:/opt/mashery/migration/
```

- Create deployment folder and extract application tar.

```
$ cd /op/mashery/migration
$ tar -xvzf tml-token-migration-v1.0.0.tar.gz
```

- Edit config.properties:

```
$ cd tml-token-migration-v1.0.0/conf
$ vi config.properties
```

Note: See the section [OAuth Token Migration Configuration Properties](#) for details of required configurations.

- Copy the exported data.tar archive to Cluster Manager container in order for this to be migrated to destination (Cassandra).

```
$ docker cp tokens.tar b77fad139de0:/opt/mashery/migration/
$ tar -xvzf tokens.tar
```

GCP Kubernetes

- Log in to the terminal from where you can access the pods.

Platform

Steps

Cluster

2. Once the cluster is up, install the migration utility:

a. Identify the Cluster Manager container name:

Identify Mashery Local CM Container				
\$ kubectl get pods				
NAME	READY	STATUS	RESTARTS	AGE
cache-deploy-0-9f66654b8-775vs	1/1	Running	0	7m
cass-set-0-0	1/1	Running	0	8m
cm-deploy-0-6fcf4c44d-fv8vf	1/1	Running	0	7m
log-set-0-0	1/1	Running	0	7m
mysql-set-0-0	1/1	Running	0	7m
tm-deploy-0-66c55846b8-4nq4r	1/1	Running	0	7m

b. Copy the migration utility tar to Cluster Manager container:

```
kubectl cp tml-token-migration-v1.0.0.tar.gz cm-deploy-0-6fcf4c44d-fv8vf:/opt/mashery/migration/
```

c. Extract the tar and write/modify suitable configuration properties in tml-token-migration-v1.0.0/conf/config.properties:

```
$ tar -xvzf tml-token-migration-v1.0.0.tar.gz
$ cd tml-token-migration-v1.0.0/conf
$ vi config.properties
```

d. Edit config.properties:

```
$ cd tml-token-migration-v1.0.0/conf
$ vi config.properties
```

Note: See the section [OAuth Token Migration Configuration Properties](#) for details of required configurations.

e. Copy the exported data.tar archive to Cluster Manager container in order for this to be migrated to the destination (Cassandra):

Platform

Steps

```
$ kubectl cp tokens.tar cm-deploy-0-6fcf4c44d-
fv8vf:/opt/mashery/migration/
$ tar -xvzf tokens.tar
```

GCP Docker
Swarm Cluster

Once the cluster is up, install the migration utility:

1. Identify the Cluster Manager container name:

<pre>\$ eval "\$(docker-machine env node-1)" \$ docker ps</pre>						
CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	
e8981a04607b	tml-tm:v5.1.0.86	"/usr/local/bin/tm-e..."	21 hours ago	Up 21 hours	22/tcp, 80/tcp, 443/tcp,	
dfeee8e46a12	tml-sql:v5.1.0.86	"/usr/local/bin/sql-..."	21 hours ago	Up 21 hours	3306/tcp	
b77fad139de0	tml-cm:v5.1.0.86	"/usr/local/bin/cm-e..."	21 hours ago	Up 21 hours		

Note: If the Cluster Manager container is running on a different node, repeat the previous step for every node (Master, etc.).

2. Repeat the previous step, with this change in the command (kubectl should be replaced with docker), for example:

```
docker cp tml-token-migration-v1.0.0.tar.gz
b77fad139de0:/opt/mashery/migration/
```

3. Create deployment folder and extract application tar.

```
$ cd /op/mashery/migration
$ tar -xvzf tml-token-migration-v1.0.0.tar.gz
```

4. Edit config.properties:

```
$ cd tml-token-migration-v1.0.0/conf
$ vi config.properties
```

Note: See the section [OAuth Token Migration Configuration Properties](#) for details of required configurations.

Platform	Steps																																																						
	5. Copy the exported data.tar archive to Cluster Manager container in order for this to be migrated to the destination (Cassandra): <pre>\$ docker cp tokens.tar b77fad139de0:/opt/mashery/migration/ \$ tar -xvzf tokens.tar</pre>																																																						
Azure Kubernetes Cluster	1. Log in to the terminal from where you can access the pods. 2. Once the cluster is up, install the migration utility: a. Identify the Cluster Manager container name: <table><tr><th colspan="6">Identify Mashery Local CM Container</th></tr><tr><td colspan="6"><pre>\$ kubectl get pods</pre></td></tr><tr><th>NAME</th><th>READY</th><th>STATUS</th><th>RESTARTS</th><th>AGE</th><th></th></tr><tr><td>cache-deploy-0-9f66654b8-775vs</td><td>1/1</td><td>Running</td><td>0</td><td>7m</td><td></td></tr><tr><td>cass-set-0-0</td><td>1/1</td><td>Running</td><td>0</td><td>8m</td><td></td></tr><tr><td>cm-deploy-0-6fcf4c44d-fv8vf</td><td>1/1</td><td>Running</td><td>0</td><td>7m</td><td></td></tr><tr><td>log-set-0-0</td><td>1/1</td><td>Running</td><td>0</td><td>7m</td><td></td></tr><tr><td>mysql-set-0-0</td><td>1/1</td><td>Running</td><td>0</td><td>7m</td><td></td></tr><tr><td>tm-deploy-0-66c55846b8-4nq4r</td><td>1/1</td><td>Running</td><td>0</td><td>7m</td><td></td></tr></table> b. Copy the migration utility tar to Cluster Manager container: <pre>kubectl cp tml-token-migration-v1.0.0.tar.gz cm-deploy-0-6fcf4c44d-fv8vf:/opt/mashery/migration/</pre> c. Extract the tar and write/modify suitable configuration properties in tml-token-migration-v1.0.0/conf/config.properties: <pre>\$ tar -xvzf tml-token-migration-v1.0.0.tar.gz \$ cd tml-token-migration-v1.0.0/conf \$ vi config.properties</pre> d. Edit config.properties: <pre>\$ cd tml-token-migration-v1.0.0/conf \$ vi config.properties</pre>	Identify Mashery Local CM Container						<pre>\$ kubectl get pods</pre>						NAME	READY	STATUS	RESTARTS	AGE		cache-deploy-0-9f66654b8-775vs	1/1	Running	0	7m		cass-set-0-0	1/1	Running	0	8m		cm-deploy-0-6fcf4c44d-fv8vf	1/1	Running	0	7m		log-set-0-0	1/1	Running	0	7m		mysql-set-0-0	1/1	Running	0	7m		tm-deploy-0-66c55846b8-4nq4r	1/1	Running	0	7m	
Identify Mashery Local CM Container																																																							
<pre>\$ kubectl get pods</pre>																																																							
NAME	READY	STATUS	RESTARTS	AGE																																																			
cache-deploy-0-9f66654b8-775vs	1/1	Running	0	7m																																																			
cass-set-0-0	1/1	Running	0	8m																																																			
cm-deploy-0-6fcf4c44d-fv8vf	1/1	Running	0	7m																																																			
log-set-0-0	1/1	Running	0	7m																																																			
mysql-set-0-0	1/1	Running	0	7m																																																			
tm-deploy-0-66c55846b8-4nq4r	1/1	Running	0	7m																																																			

Platform

Steps

Note: See the section [OAuth Token Migration Configuration Properties](#) for details of required configurations.

- e. Copy the exported data.tar archive to Cluster Manager container in order for this to be migrated to the destination (Cassandra):

```
$ kubectl cp tokens.tar cm-deploy-0-6fcf4c44d-
fv8vf:/opt/mashery/migration/
$ tar -xvzf tokens.tar
```

Azure Docker
Swarm Cluster

Once the cluster is up, install the migration utility:

1. Identify the Cluster Manager container name:

<pre>\$ eval "\$(docker-machine env node-1)" \$ docker ps</pre>						
CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	
e8981a04607b	tml-tm:v5.1.0.86	"/usr/local/bin/tm-e..."	21 hours ago	Up 21 hours	22/tcp, 80/tcp, 443/tcp,	
dfeee8e46a12	tml-sql:v5.1.0.86	"/usr/local/bin/sql-..."	21 hours ago	Up 21 hours	3306/tcp	
b77fad139de0	tml-cm:v5.1.0.86	"/usr/local/bin/cm-e..."	21 hours ago	Up 21 hours		

Note: If the Cluster Manager container is running on a different node, repeat the previous step for every node (Master, etc.).

2. Repeat the previous step, with this change in the command (kubectl should be replaced with docker), for example:

```
docker cp tml-token-migration-v1.0.0.tar.gz
b77fad139de0:/opt/mashery/migration/
```

3. Create deployment folder and extract application tar.

```
$ cd /op/mashery/migration
$ tar -xvzf tml-token-migration-v1.0.0.tar.gz
```

Platform	Steps																																																						
	4. Edit config.properties: <pre>\$ cd tml-token-migration-v1.0.0/conf \$ vi config.properties</pre> Note: See the section OAuth Token Migration Configuration Properties for details of required configurations. 5. Copy the exported data.tar archive to Cluster Manager container in order for this to be migrated to the destination (Cassandra): <pre>\$ docker cp tokens.tar b77fad139de0:/opt/mashery/migration/ \$ tar -xvzf tokens.tar</pre>																																																						
On-premises Kubernetes Cluster	1. Log in to the terminal from where you can access the pods. 2. Once the cluster is up, install the migration utility: a. Identify the Cluster Manager container name: <table><tr><th colspan="6">Identify Mashery Local CM Container</th></tr><tr><td colspan="6"><pre>\$ kubectl get pods</pre></td></tr><tr><td>NAME</td><td>READY</td><td>STATUS</td><td>RESTARTS</td><td>AGE</td><td></td></tr><tr><td>cache-deploy-0-9f66654b8-775vs</td><td>1/1</td><td>Running</td><td>0</td><td>7m</td><td></td></tr><tr><td>cass-set-0-0</td><td>1/1</td><td>Running</td><td>0</td><td>8m</td><td></td></tr><tr><td>cm-deploy-0-6fcf4c44d-fv8vf</td><td>1/1</td><td>Running</td><td>0</td><td>7m</td><td></td></tr><tr><td>log-set-0-0</td><td>1/1</td><td>Running</td><td>0</td><td>7m</td><td></td></tr><tr><td>mysql-set-0-0</td><td>1/1</td><td>Running</td><td>0</td><td>7m</td><td></td></tr><tr><td>tm-deploy-0-66c55846b8-4nq4r</td><td>1/1</td><td>Running</td><td>0</td><td>7m</td><td></td></tr></table> b. Copy the migration utility tar to Cluster Manager container: <pre>kubectl cp tml-token-migration-v1.0.0.tar.gz cm-deploy-0-6fcf4c44d-fv8vf:/opt/mashery/migration/</pre> c. Extract the tar and write/modify suitable configuration properties in tml-token-migration-v1.0.0/conf/config.properties:	Identify Mashery Local CM Container						<pre>\$ kubectl get pods</pre>						NAME	READY	STATUS	RESTARTS	AGE		cache-deploy-0-9f66654b8-775vs	1/1	Running	0	7m		cass-set-0-0	1/1	Running	0	8m		cm-deploy-0-6fcf4c44d-fv8vf	1/1	Running	0	7m		log-set-0-0	1/1	Running	0	7m		mysql-set-0-0	1/1	Running	0	7m		tm-deploy-0-66c55846b8-4nq4r	1/1	Running	0	7m	
Identify Mashery Local CM Container																																																							
<pre>\$ kubectl get pods</pre>																																																							
NAME	READY	STATUS	RESTARTS	AGE																																																			
cache-deploy-0-9f66654b8-775vs	1/1	Running	0	7m																																																			
cass-set-0-0	1/1	Running	0	8m																																																			
cm-deploy-0-6fcf4c44d-fv8vf	1/1	Running	0	7m																																																			
log-set-0-0	1/1	Running	0	7m																																																			
mysql-set-0-0	1/1	Running	0	7m																																																			
tm-deploy-0-66c55846b8-4nq4r	1/1	Running	0	7m																																																			

Platform	Steps
	<pre>\$ tar -xvzf tml-token-migration-v1.0.0.tar.gz \$ cd tml-token-migration-v1.0.0/conf \$ vi config.properties</pre> d. Edit config.properties: <pre>\$ cd tml-token-migration-v1.0.0/conf \$ vi config.properties</pre> Note: See the section OAuth Token Migration Configuration Properties for details of required configurations. e. Copy the exported data.tar archive to Cluster Manager container in order for this to be migrated to the destination (Cassandra): <pre>\$ kubectl cp tokens.tar cm-deploy-0-6fcf4c44d-fv8vf:/opt/mashery/migration/ \$ tar -xvzf tokens.tar</pre>

On-premises Docker Swarm Cluster

Once the cluster is up, install the migration utility:

1. Identify the Cluster Manager container name:

Identify Mashery Local CM Container						
<pre>\$ eval "\$(docker-machine env node-1)" \$ docker ps</pre>						
CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	
e8981a04607b	tml-tm:v5.1.0.86	"/usr/local/bin/tm-e..."	21 hours ago	Up 21 hours	22/tcp, 80/tcp, 443/tcp,	
dfeee8e46a12	tml-sql:v5.1.0.86	"/usr/local/bin/sql-..."	21 hours ago	Up 21 hours	3306/tcp	
b77fad139de0	tml-cm:v5.1.0.86	"/usr/local/bin/cm-e..."	21 hours ago	Up 21 hours		

Note: If the Cluster Manager container is running on a different node, repeat the previous step for every node (Master, etc.).

2. Repeat the previous step, with this change in the command

Platform	Steps
	(kubectl should be replaced with docker), for example: <pre>docker cp tml-token-migration-v1.0.0.tar.gz b77fad139de0:/opt/mashery/migration/</pre>
	3. Create deployment folder and extract application tar. <pre>\$ cd /op/mashery/migration \$ tar -xvzf tml-token-migration-v1.0.0.tar.gz</pre>
	4. Edit config.properties: <pre>\$ cd tml-token-migration-v1.0.0/conf \$ vi config.properties</pre> Note: See the section OAuth Token Migration Configuration Properties for details of required configurations.
	5. Copy the exported data.tar archive to Cluster Manager container in order for this to be migrated to the destination (Cassandra): <pre>\$ docker cp tokens.tar b77fad139de0:/opt/mashery/migration/ \$ tar -xvzf tokens.tar</pre>

OAuth Token Migration Configuration Properties

The configuration file for OAuth Token Migration is located in tml-token-migration-v1.0.0/conf/config.properties.

The configuration file need to be modified depending upon the source and destination combination. It could be one among following:

MySQL to Cassandra

```
config.source.type=MYSQL
config.dest.type=CASSANDRA
config.source.mysql.pwd=<REPLACE_WITH_SOURCE_MYSQL_PASSWORD>
config.source.mysql.user=<REPLACE_WITH_SOURCE_MYSQL_USER>
config.source.mysql.port=<REPLACE_WITH_SOURCE_MYSQL_PORT>
config.source.mysql.host=<REPLACE_WITH_SOURCE_MYSQL_HOST>
config.dest.cass.host=<REPLACE_WITH_CASSANDRA_HOST>
config.dest.cass.port=<REPLACE_WITH_CASSANDRA_PORT>
```

MySQL to FILE

```
config.source.type=MYSQL
config.dest.type=FILE
config.source.mysql.pwd=<REPLACE_WITH_SOURCE_MYSQL_PASSWORD>
config.source.mysql.user=<REPLACE_WITH_SOURCE_MYSQL_USER>
config.source.mysql.port=<REPLACE_WITH_SOURCE_MYSQL_PORT>
config.source.mysql.host=<REPLACE_WITH_SOURCE_MYSQL_HOST>
config.dest.file.directory=<REPLACE_WITH_DEST_FILE_DIRECTORY>
```

FILE to Cassandra

```
config.source.type=FILE
config.dest.type=CASSANDRA
config.source.file.directory=<REPLACE_WITH_DEST_FILE_DIRECTORY>
config.dest.cass.host=<REPLACE_WITH_CASSANDRA_HOST>
config.dest.cass.port=<REPLACE_WITH_CASSANDRA_PORT>
```

Cassandra to FILE

```
config.source.type=CASSANDRA
config.dest.type=FILE
config.source.cass.host=<REPLACE_WITH_CASSANDRA_HOST>
config.source.cass.port=<REPLACE_WITH_CASSANDRA_PORT>
config.dest.file.directory=<REPLACE_WITH_DEST_FILE_DIRECTORY>
```

FILE to MySQL

```
config.source.type=FILE
config.dest.type=MYSQL
config.source.file.directory=<REPLACE_WITH_DEST_FILE_DIRECTORY>

config.dest.mysql.pwd=<REPLACE_WITH_SOURCE_MYSQL_PASSWORD>
config.dest.mysql.user=<REPLACE_WITH_SOURCE_MYSQL_USER>
config.dest.mysql.port=<REPLACE_WITH_SOURCE_MYSQL_PORT>
config.dest.mysql.host=<REPLACE_WITH_SOURCE_MYSQL_HOST>
```



Note: In order to put the Cassandra hostname or IP address in the configuration property file, you'll need to get it from the cluster.

Docker Swarm Cluster

Get Cassandra hostname or IP address
\$ docker ps

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
7fb15fa9a55c	tml-cache:v5.1.0.293	"/usr/local/bin/cach..."	46 hours ago	Up 46 hours	11211-11216/tcp	cachestack_cache.1.datfwnvhlhtytlizkov7u3j
f454a5b50bc3	tml-sql:v5.1.0.293	"/usr/local/bin/sql-..."	46 hours ago	Up 46 hours	3306/tcp	sqlstack_sql.1.xw1ht5lqwpeha372chfckyqm3
2435340b854f	tml-log:v5.1.0.293	"/usr/local/bin/log-..."	46 hours ago	Up 46 hours	24023/tcp, 24220/tcp, 24224/tcp	logstack_log.1.6o8mgpohak63201zjfdszng3s
c80a84800d5d	tml-api:v5.1.0.293	"/usr/local/bin/api-..."	46 hours ago	Up 46 hours		cmstack_tmlapi.1.fjmmaurkj34mdnvs30l0ytype
b33a5bfe979f	tml-nosql:v5.1.0.293	"/usr/local/bin/nosq..."	46 hours ago	Up 46 hours (healthy)	7000-7001/tcp, 7199/tcp, 9042/tcp, 9160/tcp	nosqlstack_nosql_seed.1.reza1aic3ktnbov2jj8gutms6

cassandra host define in config.properties

```
config.dest.cass.host=b33a5bfe979f
```

Kubernetes Cluster

**Get Cassandra hostname or IP address**

```
$ kubectl get pods
```

NAME	READY	STATUS	RESTARTS	AGE
cache-set-0-0	1/1	Running	0	2d
cache-set-0-1	1/1	Running	0	2d
cass-set-0-0	1/1	Running	0	2d
cass-set-0-1	1/1	Running	0	2d
cm-deploy-0-7d864c459f-6scdg	2/2	Running	0	2d
cm-deploy-1-774c6f94c9-g64nn	2/2	Running	0	2d
log-set-0-0	1/1	Running	0	2d
log-set-1-1	1/1	Running	0	2d
mysql-set-0-0	1/1	Running	0	2d
tm-deploy-0-5d6f55b566-cnqdj	1/1	Running	0	2d
tm-deploy-1-65bd4888c6-vz69q	1/1	Running	0	2d

In a multizone Kubernetes setup, the Cassandra node host property in `config.properties` should be defined in a fully-qualified way:

```
config.dest.cass.host=cass-set-0-0.cass-svc-0.default.svc.cluster.local
```

Running the OAuth Token Migration Utility

To run the OAuth Token Migration utility:

1. Go to the directory where the utility is deployed, for example: `/opt/mashery/migration`.
2. Go to the application bin directory.
3. Run the startup command.

For example:

```
[root@88dfda2eecab builder]# cd /opt/mashery/migration/
[root@88dfda2eecab migration]# cd tml-token-migration-v1.0.0/bin
[root@88dfda2eecab bin]# ./tml-token-migration ../conf/config.properties
```

```
[INFO ] 2019-03-17 07:23:41.403 [main] c.m.tml.token.migration.Launcher - Migration program
starting now with configuration file: ../conf/config.properties
[INFO ] 2019-03-17 07:23:41.413 [pool-2-thread-2] c.m.t.t.m.s.job.BatchReadJob - Starting read
thread now
[INFO ] 2019-03-17 07:23:41.874 [pool-2-thread-2] c.m.t.t.m.s.job.BatchReadJob - Reading data
between interval : Fri Feb 22 10:00:02 PST 2019 - Mon Feb 25 07:05:02 PST 2019
[INFO ] 2019-03-17 07:23:41.879 [pool-2-thread-2] c.m.t.t.m.s.FilePeristenceManager - Reading
File data between interval : Fri Feb 22 10:00:02 PST 2019 - Mon Feb 25 07:05:02 PST 2019
[INFO ] 2019-03-17 07:23:41.897 [pool-2-thread-2] c.m.t.t.m.s.FilePeristenceManager - Read of
File source tokens finished, total no of file tokens fetched 26550
...
..
```

If the path to config.properties is not provided, it will use a default path, for example, tml-token-migration-v1.0.0/conf/config.properties.

```
[root@64bfbb9d9706 bin]# ./tml-token-migration
[INFO ] 2019-04-05 13:13:52.265 [main] c.m.t.t.m.MigrationConfiguration - Migration is configured
for SOURCE = CASSANDRA
[INFO ] 2019-04-05 13:13:52.274 [main] c.m.t.t.m.MigrationConfiguration - Migration is configured
for DESTINATION = FILE
[INFO ] 2019-04-05 13:13:52.347 [main] c.m.tml.token.migration.Launcher - Migration program
starting now with configuration file: ../conf/config.properties
...
```

Running the Utility for the First Time

Upon running the utility for the first time, it will try to migrate all tokens (oldest updated token date to latest updated token date).

Running the Utility for the Second Time

Once all tokens are migrated from Source to Destination, just before switching over the load balancer to point to the Destination cluster, the migration utility needs to be run once again to migrate those tokens which were created/updated during the first migration timeframe window.

Running the Utility for the Third Time

Now after second time migration run, load balancers are switched over to the Destination cluster, then the migration utility needs to be run once again to migrate any leftover tokens which were created/updated during the second run migration timeframe window.

The three runs will ensure that all tokens are migrated from source to destination.

Failover Run in Case Process was Stopped during any Migration Run

If the process gets killed or stopped on its own in between the migration process, it will start from where it finished in the last run.

For example:

- In the first run, the utility started migrating all the tokens between time interval, 22-01-2019 02:00:00 to 15-03-2019 04:45:10 (both inclusive).
- After some time, the process got killed or stopped on its own due to some reason, at that time, it migrated tokens till updated time, 01-02-2019 03:40:00. This time entry is written in the Journal file (location at /opt/mashery/migration/tml-token-migration-v1.0.0/data/MigrationJournal.txt)

```
[root@64bfbb9d9706 bin]# cat ../data/MigrationJournal.txt
1548182063000 Migrate-Token 600000 Fri Apr 05 15:08:28 UTC 2019 Fri Apr 05 16:28:39
UTC 2019
```

- So in the subsequent run, it will try to pick tokens between 01-02-2019 03:40:00 and 15-03-2019 04:45:10 (both inclusive).

Token Migration Notes

- Create separate data directory for each run if doing FILE based migration.
- Change destination file directory (created during MySQL to FILE data export) to source file directory when running migration from FILE to CASSANDRA.
- Clear journal entry of last run (FILE to CASSANDRA import) when executing next migration run from FILE to CASSANDRA.

Checkpoint File

After every run, the utility creates an entry in a checkpoint file named MigrationJournal.txt. It essentially records the updated timestamp of last token which was written to destination.

Format (entries separated by tab):

Last_written_record_updated_timestamp	Id_of_last_token	Source	No_of_records_written
Batch_start_time	Batch_end_time		

For example:

1548182063000	234212	Migrate-Token	600000	Fri Apr 05
15:08:28 UTC 2019	Fri Apr 05 16:28:39 UTC 2019			

This entry will be used in subsequent runs to load the records from the point it stopped writing.

Sample Run Logs

```
root@d2d3421x1332:bin$ ./tml-token-migration ../conf/config.properties
[INFO ] 2019-04-01 18:58:10.283 [main] c.m.t.t.m.MigrationConfiguration - Migration is configured
for SOURCE = MYSQL
[INFO ] 2019-04-01 18:58:10.287 [main] c.m.t.t.m.MigrationConfiguration - Migration is configured
for DESTINATION = CASSANDRA
[INFO ] 2019-04-01 18:58:10.329 [main] c.m.tml.token.migration.Launcher - Migration program
starting now with configuration file: ../conf/config.properties
[INFO ] 2019-04-01 18:58:10.894 [pool-2-thread-2] c.m.t.t.m.s.job.BatchReadJob - Starting read
thread now
[INFO ] 2019-04-01 18:58:11.397 [pool-2-thread-2] c.m.t.t.m.s.job.BatchReadJob - Reading data
between interval : 2019-01-22 23:47:41.0 - 2019-01-23 03:56:07.0
[INFO ] 2019-04-01 18:58:11.397 [pool-2-thread-2] c.m.t.t.m.s.MySQLPersistenceManager -
Reading MySQL data between interval : 2019-01-22 23:47:41.0 - 2019-01-23 03:56:07.0
[INFO ] 2019-04-01 18:58:13.802 [pool-2-thread-2] c.m.t.t.m.s.MySQLPersistenceManager - Read
of MySQL source tokens finished, total no of file tokens fetched 211991
[INFO ] 2019-04-01 18:58:13.803 [pool-2-thread-2] c.m.t.t.m.s.MySQLPersistenceManager - Read
of MySQL source tokens completed...
[INFO ] 2019-04-01 18:58:13.803 [pool-2-thread-2] c.m.t.t.m.s.job.BatchReadJob - Fetched data
from source, size of 211991 no of records
[INFO ] 2019-04-01 18:58:13.833 [pool-2-thread-1] c.m.t.t.m.s.job.MigrationJob - Submitted Batch
read Job to executor
[INFO ] 2019-04-01 18:58:13.833 [pool-2-thread-1] c.m.t.t.m.s.job.MigrationJob - Submitted Batch
write Job with 1 threads
[INFO ] 2019-04-01 18:58:14.823 [pool-2-thread-2] c.m.t.t.m.s.job.BatchWriteJob - Starting write
thread now
[INFO ] 2019-04-01 18:58:14.974 [pool-2-thread-2] c.m.t.t.m.s.job.BatchWriteJob - Total 211991
found to be migrated, migrating now.
[INFO ] 2019-04-01 18:58:20.917 [pool-2-thread-2] c.m.t.t.m.s.CassandraPersistenceManager -
Cassandra Write Batch commit started at: 2019-04-01 18:58:20.917 and finished at: 2019-04-01
18:58:20.917. Total writes so far 5000
```


[INFO] 2019-04-01 18:58:24.972 [pool-2-thread-2] c.m.t.t.m.s.CassandraPersistenceManager - Cassandra Write Batch commit started at: 2019-04-01 18:58:24.971 and finished at: 2019-04-01 18:58:24.971. Total writes so far 10000

[INFO] 2019-04-01 18:58:28.889 [pool-2-thread-2] c.m.t.t.m.s.CassandraPersistenceManager - Cassandra Write Batch commit started at: 2019-04-01 18:58:28.889 and finished at: 2019-04-01 18:58:28.889. Total writes so far 15000

[INFO] 2019-04-01 18:58:32.999 [pool-2-thread-2] c.m.t.t.m.s.CassandraPersistenceManager - Cassandra Write Batch commit started at: 2019-04-01 18:58:32.999 and finished at: 2019-04-01 18:58:32.999. Total writes so far 20000

[INFO] 2019-04-01 18:58:37.974 [pool-2-thread-2] c.m.t.t.m.s.CassandraPersistenceManager - Cassandra Write Batch commit started at: 2019-04-01 18:58:37.974 and finished at: 2019-04-01 18:58:37.974. Total writes so far 25000

[INFO] 2019-04-01 18:58:42.119 [pool-2-thread-2] c.m.t.t.m.s.CassandraPersistenceManager - Cassandra Write Batch commit started at: 2019-04-01 18:58:42.119 and finished at: 2019-04-01 18:58:42.119. Total writes so far 30000

[INFO] 2019-04-01 18:58:46.175 [pool-2-thread-2] c.m.t.t.m.s.CassandraPersistenceManager - Cassandra Write Batch commit started at: 2019-04-01 18:58:46.175 and finished at: 2019-04-01 18:58:46.175. Total writes so far 35000

[INFO] 2019-04-01 18:58:50.516 [pool-2-thread-2] c.m.t.t.m.s.CassandraPersistenceManager - Cassandra Write Batch commit started at: 2019-04-01 18:58:50.516 and finished at: 2019-04-01 18:58:50.516. Total writes so far 40000

[INFO] 2019-04-01 18:58:55.163 [pool-2-thread-2] c.m.t.t.m.s.CassandraPersistenceManager - Cassandra Write Batch commit started at: 2019-04-01 18:58:55.163 and finished at: 2019-04-01 18:58:55.163. Total writes so far 45000

[INFO] 2019-04-01 18:58:59.345 [pool-2-thread-2] c.m.t.t.m.s.CassandraPersistenceManager - Cassandra Write Batch commit started at: 2019-04-01 18:58:59.345 and finished at: 2019-04-01 18:58:59.345. Total writes so far 50000

[INFO] 2019-04-01 18:59:03.493 [pool-2-thread-2] c.m.t.t.m.s.CassandraPersistenceManager - Cassandra Write Batch commit started at: 2019-04-01 18:59:03.492 and finished at: 2019-04-01 18:59:03.492. Total writes so far 55000

[INFO] 2019-04-01 18:59:07.581 [pool-2-thread-2] c.m.t.t.m.s.CassandraPersistenceManager - Cassandra Write Batch commit started at: 2019-04-01 18:59:07.581 and finished at: 2019-04-01 18:59:07.581. Total writes so far 60000

[INFO] 2019-04-01 18:59:11.670 [pool-2-thread-2] c.m.t.t.m.s.CassandraPersistenceManager - Cassandra Write Batch commit started at: 2019-04-01 18:59:11.67 and finished at: 2019-04-01 18:59:11.67. Total writes so far 65000

[INFO] 2019-04-01 18:59:15.822 [pool-2-thread-2] c.m.t.t.m.s.CassandraPersistenceManager - Cassandra Write Batch commit started at: 2019-04-01 18:59:15.822 and finished at: 2019-04-01 18:59:15.822. Total writes so far 70000

[INFO] 2019-04-01 18:59:19.963 [pool-2-thread-2] c.m.t.t.m.s.CassandraPersistenceManager - Cassandra Write Batch commit started at: 2019-04-01 18:59:19.963 and finished at: 2019-04-01 18:59:19.963. Total writes so far 75000

[INFO] 2019-04-01 18:59:24.156 [pool-2-thread-2] c.m.t.t.m.s.CassandraPersistenceManager - Cassandra Write Batch commit started at: 2019-04-01 18:59:24.156 and finished at: 2019-04-01 18:59:24.156. Total writes so far 80000

[INFO] 2019-04-01 18:59:29.087 [pool-2-thread-2] c.m.t.t.m.s.CassandraPersistenceManager - Cassandra Write Batch commit started at: 2019-04-01 18:59:29.086 and finished at: 2019-04-01 18:59:29.086. Total writes so far 85000

[INFO] 2019-04-01 18:59:35.411 [pool-2-thread-2] c.m.t.t.m.s.CassandraPersistenceManager - Cassandra Write Batch commit started at: 2019-04-01 18:59:35.411 and finished at: 2019-04-01 18:59:35.411. Total writes so far 90000

[INFO] 2019-04-01 18:59:39.609 [pool-2-thread-2] c.m.t.t.m.s.CassandraPersistenceManager - Cassandra Write Batch commit started at: 2019-04-01 18:59:39.609 and finished at: 2019-04-01 18:59:39.609. Total writes so far 95000

[INFO] 2019-04-01 18:59:43.791 [pool-2-thread-2] c.m.t.t.m.s.CassandraPersistenceManager - Cassandra Write Batch commit started at: 2019-04-01 18:59:43.79 and finished at: 2019-04-01 18:59:43.79. Total writes so far 100000

[INFO] 2019-04-01 18:59:48.208 [pool-2-thread-2] c.m.t.t.m.s.CassandraPersistenceManager - Cassandra Write Batch commit started at: 2019-04-01 18:59:48.208 and finished at: 2019-04-01 18:59:48.208. Total writes so far 105000

[INFO] 2019-04-01 18:59:52.519 [pool-2-thread-2] c.m.t.t.m.s.CassandraPersistenceManager - Cassandra Write Batch commit started at: 2019-04-01 18:59:52.519 and finished at: 2019-04-01 18:59:52.519. Total writes so far 110000

[INFO] 2019-04-01 18:59:56.775 [pool-2-thread-2] c.m.t.t.m.s.CassandraPersistenceManager - Cassandra Write Batch commit started at: 2019-04-01 18:59:56.775 and finished at: 2019-04-01 18:59:56.775. Total writes so far 115000

[INFO] 2019-04-01 19:00:01.060 [pool-2-thread-2] c.m.t.t.m.s.CassandraPersistenceManager - Cassandra Write Batch commit started at: 2019-04-01 19:00:01.06 and finished at: 2019-04-01 19:00:01.06. Total writes so far 120000

[INFO] 2019-04-01 19:00:05.392 [pool-2-thread-2] c.m.t.t.m.s.CassandraPersistenceManager - Cassandra Write Batch commit started at: 2019-04-01 19:00:05.392 and finished at: 2019-04-01 19:00:05.392. Total writes so far 125000

[INFO] 2019-04-01 19:00:10.743 [pool-2-thread-2] c.m.t.t.m.s.CassandraPersistenceManager - Cassandra Write Batch commit started at: 2019-04-01 19:00:10.743 and finished at: 2019-04-01 19:00:10.743. Total writes so far 130000

[INFO] 2019-04-01 19:00:15.069 [pool-2-thread-2] c.m.t.t.m.s.CassandraPersistenceManager - Cassandra Write Batch commit started at: 2019-04-01 19:00:15.069 and finished at: 2019-04-01 19:00:15.069. Total writes so far 135000

[INFO] 2019-04-01 19:00:21.381 [pool-2-thread-2] c.m.t.t.m.s.CassandraPersistenceManager - Cassandra Write Batch commit started at: 2019-04-01 19:00:21.381 and finished at: 2019-04-01 19:00:21.381. Total writes so far 140000

[INFO] 2019-04-01 19:00:25.702 [pool-2-thread-2] c.m.t.t.m.s.CassandraPersistenceManager - Cassandra Write Batch commit started at: 2019-04-01 19:00:25.702 and finished at: 2019-04-01 19:00:25.702. Total writes so far 145000

[INFO] 2019-04-01 19:00:30.099 [pool-2-thread-2] c.m.t.t.m.s.CassandraPersistenceManager - Cassandra Write Batch commit started at: 2019-04-01 19:00:30.098 and finished at: 2019-04-01 19:00:30.098. Total writes so far 150000

[INFO] 2019-04-01 19:00:34.516 [pool-2-thread-2] c.m.t.t.m.s.CassandraPersistenceManager - Cassandra Write Batch commit started at: 2019-04-01 19:00:34.516 and finished at: 2019-04-01 19:00:34.516. Total writes so far 155000

```

[INFO ] 2019-04-01 19:00:39.065 [pool-2-thread-2] c.m.t.t.m.s.CassandraPersistenceManager -
Cassandra Write Batch commit started at: 2019-04-01 19:00:39.065 and finished at: 2019-04-01
19:00:39.065. Total writes so far 160000
[INFO ] 2019-04-01 19:00:43.510 [pool-2-thread-2] c.m.t.t.m.s.CassandraPersistenceManager -
Cassandra Write Batch commit started at: 2019-04-01 19:00:43.51 and finished at: 2019-04-01
19:00:43.51. Total writes so far 165000
[INFO ] 2019-04-01 19:00:48.162 [pool-2-thread-2] c.m.t.t.m.s.CassandraPersistenceManager -
Cassandra Write Batch commit started at: 2019-04-01 19:00:48.162 and finished at: 2019-04-01
19:00:48.162. Total writes so far 170000
[INFO ] 2019-04-01 19:00:52.686 [pool-2-thread-2] c.m.t.t.m.s.CassandraPersistenceManager -
Cassandra Write Batch commit started at: 2019-04-01 19:00:52.686 and finished at: 2019-04-01
19:00:52.686. Total writes so far 175000
[INFO ] 2019-04-01 19:00:59.547 [pool-2-thread-2] c.m.t.t.m.s.CassandraPersistenceManager -
Cassandra Write Batch commit started at: 2019-04-01 19:00:59.546 and finished at: 2019-04-01
19:00:59.547. Total writes so far 180000
[INFO ] 2019-04-01 19:01:04.368 [pool-2-thread-2] c.m.t.t.m.s.CassandraPersistenceManager -
Cassandra Write Batch commit started at: 2019-04-01 19:01:04.368 and finished at: 2019-04-01
19:01:04.368. Total writes so far 185000
[INFO ] 2019-04-01 19:01:09.093 [pool-2-thread-2] c.m.t.t.m.s.CassandraPersistenceManager -
Cassandra Write Batch commit started at: 2019-04-01 19:01:09.093 and finished at: 2019-04-01
19:01:09.093. Total writes so far 190000
[INFO ] 2019-04-01 19:01:13.832 [pool-2-thread-2] c.m.t.t.m.s.CassandraPersistenceManager -
Cassandra Write Batch commit started at: 2019-04-01 19:01:13.831 and finished at: 2019-04-01
19:01:13.831. Total writes so far 195000
[INFO ] 2019-04-01 19:01:22.035 [pool-2-thread-2] c.m.t.t.m.s.CassandraPersistenceManager -
Cassandra Write Batch commit started at: 2019-04-01 19:01:22.035 and finished at: 2019-04-01
19:01:22.035. Total writes so far 200000
[INFO ] 2019-04-01 19:01:26.945 [pool-2-thread-2] c.m.t.t.m.s.CassandraPersistenceManager -
Cassandra Write Batch commit started at: 2019-04-01 19:01:26.945 and finished at: 2019-04-01
19:01:26.945. Total writes so far 205000
[INFO ] 2019-04-01 19:01:32.132 [pool-2-thread-2] c.m.t.t.m.s.CassandraPersistenceManager -
Cassandra Write Batch commit started at: 2019-04-01 19:01:32.132 and finished at: 2019-04-01
19:01:32.132. Total writes so far 210000
[INFO ] 2019-04-01 19:01:34.202 [pool-2-thread-2] c.m.t.t.m.s.CassandraPersistenceManager -
Last written Cassandra token is w32wz5q5mygu44ya59w5rrfk : Wed Jan 23 03:56:07 IST 2019
[INFO ] 2019-04-01 19:01:34.202 [pool-2-thread-2] c.m.t.t.m.s.job.BatchWriteJob - Finished write
thread now
[ERROR] 2019-04-01 19:02:34.205 [pool-2-thread-1] c.m.t.t.m.s.job.MigrationJob - Could not
terminate running Migration job cleanly, shutting down forcefully
[INFO ] 2019-04-01 19:02:34.208 [pool-2-thread-1] c.m.t.t.m.s.job.MigrationJob - Migration job
completed, shutting down now
[INFO ] 2019-04-01 19:02:34.208 [main] c.m.t.t.m.service.Migrator - Migration program exiting with
status code 0
[INFO ] 2019-04-01 19:02:34.209 [Thread-2] c.m.t.t.m.s.FileJournalService - Journal service got
last written record updated: Wed Jan 23 03:56:07 IST 2019

```

```
[INFO ] 2019-04-01 19:02:34.210 [Thread-2] c.m.t.t.m.s.FileJournalService - File journal path is
/opt/mashery/migration/data/MigrationJournal.txt
[INFO ] 2019-04-01 19:02:34.210 [Thread-2] c.m.t.t.m.s.FileJournalService - File journal is not
existing, creating it /opt/mashery/migration/data/MigrationJournal.txt
[INFO ] 2019-04-01 19:02:34.214 [Thread-2] c.m.t.t.m.s.FileJournalService - Writing file journal
entry for Migrate-Token
root@d2d3421x1332:bin$
```

Verifying OAuth Token Migration

Once migration is complete, you can verify if the tokens were written to the destination.

For example, if migrating from MySQL to Cassandra, you can verify in a CQLSH prompt whether a specific source token is present:

1. Take a random token from source tokens file(s) or MySQL:

```
$ tail -f tokens/token_1551107102000_1554628596851.json
```

```
{
  "id": "155218",
  "updated": "1554463157000",
  "areaid": "711",
  "spkey": "fzaqf787e3pu3uqcz764uj9y",
  "mapi": "v4kf8heb7r8k2xcjyzv8xdjz",
  "accessToken": "d4uhgedp24hj8hdfqkqmexsu",
  "accessTokenType": "0",
  "grantType": "implicit",
  "scope": "scope",
  "expires": "1555428066000",
  "macKey": null,
  "macAlgorithm": "1",
  "refreshToken": null,
  "refreshTokenExpire": null,
  "userToken": null,
  "isLocal": "0"
}
```

2. Login to Cassandra container (for example, K8 environment):

```
$ kubectl exec -it cass-set-0-0 bash
```

3. Got to CQL shell prompt and find the specific token which should have been migrated:

```
$ cqlsh
Connected to Tibco Mashery Local Reference Cluster at 127.0.0.1:9042.
[cqlsh 5.0.1 | Cassandra 3.11.2 | CQL spec 3.4.4 | Native protocol v4]
Use HELP for help.
cqlsh> use oauth2;
cqlsh> select * from atokens where atoken = 'druds84xr7k7tyupvwc5gwkj';
+-----+-----+-----+-----+-----+-----+-----+-----+
| atoken | spkey | atokentype | created | expires |          |          |
| grnt   | macalgo | mackey | mapi   | scope | updated |          | utoken |
+-----+-----+-----+-----+-----+-----+-----+-----+
|       |       |          |        |        |         |         |       |
+-----+-----+-----+-----+-----+-----+-----+-----+
```

```

-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
+-----+
druds84xr7k7tyupvwc5gwkj | m9f5kyzfjrrsz6d5sfkbuw3 |      0 | 2019-01-22
19:31:54.000000+0000 | 2019-04-08 05:19:36.000000+0000 | password |      1 | null |
4c7fmwh59w3ufn7pxt92qwx9 | null | 2019-01-22 19:31:54.000000+0000 |      null

```

OAuth Token Migration FAQs

Q: How do you find the username and password for MySQL?

A: In your Local Edition installation, you can find the password for the user **mashclient** from `/etc/mashery-system.conf`:

```

"client":
{"password":"XXXXXXXXXXXXX","user":"mashclient"},

```

Q: Can the utility be run multiple times?

A: Yes. Each time it runs, it will pick all tokens created or updated after the last run of the utility. The utility maintains a journal entry of last-processed tokens.

Q: When do I need to use FILE as an intermediary to migrate tokens between Local Edition 4.1 and Local Edition 5.1?

Environments may exist where it is not possible to talk to the MySQL database from the Local Edition 5.1 Cluster Manager container. Therefore, it is advised to do the migration in 2 steps:

1. Migrate Local Edition 4.X MySQL to FILE.
2. Migrate FILE to Local Edition 5.1 Cassandra.

Q: How do you migrate tokens with the least downtime?

A: To achieve near-zero downtime:

1. Run the utility once to migrate all known tokens.
2. Run the utility again, right before the switching the traffic from Local Edition 4.x to Local Edition 5x to migrate tokens generated after the run in Step 1.
3. Run the utility right after the switch to migrate tokens generated after run in step 2.

Troubleshooting OAuth Token Migration Issues

Cassandra Write Timeout Issue

While running the Migration utility in Cloud storage (for example, AWS or GCP), after writing a number of tokens in Cassandra, the utility may sometimes give the following error and fail.

```
com.datastax.driver.core.exceptions.WriteTimeoutException: Cassandra timeout during SIMPLE
write query at consistency LOCAL_QUORUM (2 replica were required but only 1 acknowledged the
write)
    at com.datastax.driver.core.Responses$Error$1.decode(Responses.java:88)
    at com.datastax.driver.core.Responses$Error$1.decode(Responses.java:66)
    at com.datastax.driver.core.Message$ProtocolDecoder.decode(Message.java:297)
    at com.datastax.driver.core.Message$ProtocolDecoder.decode(Message.java:268)
    at io.netty.handler.codec.MessageToMessageDecoder.channelRead
(MessageToMessageDecoder.java:88)
    ... 25 common frames omitted
```

This error is caused in Cloud cluster setup, where the write operation was taking more time than the write timeout value defined in `cassandra.yml` (`write_request_timeout_in_ms`: default value 2 seconds).

In this case, run the utility again. In next migration, running it should start the write operation.

i Note: If the application does not exit after throwing the exception, press Ctrl-C. This will exit the application and will write the checkpoint file with last run details.

Cassandra Commands to Check Health of Cassandra Cluster

Check atoken table statistics to check write status:

```
[root@cass-set-0-0 builder]# nodetool tablestats -H oauth2.atokens;
Total number of tables: 57
```

```
-----
Keyspace : oauth2
  Read Count: 0
  Read Latency: NaN ms
  Write Count: 3954158
  Write Latency: 0.06952324439235862 ms
  Pending Flushes: 0
    Table: atokens
      SSTable count: 4
      Space used (live): 83.21 MiB
      Space used (total): 83.21 MiB
      Off heap memory used (total): 0 bytes
      SSTable Compression Ratio: -1.0
      Number of partitions (estimate): 112
      Memtable cell count: 143
      Memtable data size: 31.12 KiB
      Memtable off heap memory used: 0 bytes
      Memtable switch count: 41
      Local read count: 0
      Local read latency: NaN ms
      Local write count: 242672
      Local write latency: NaN ms
      Pending flushes: 0
      Percent repaired: 100.0
      Bloom filter false positives: 0
      Bloom filter false ratio: 0.00000
      Bloom filter space used: 0 bytes
      Bloom filter off heap memory used: 0 bytes
      Index summary off heap memory used: 0 bytes
      Compression metadata off heap memory used: 0 bytes
      Compacted partition minimum bytes: 0
      Compacted partition maximum bytes: 0
      Compacted partition mean bytes: 0
      Average live cells per slice (last five minutes): NaN
      Maximum live cells per slice (last five minutes): 0
      Average tombstones per slice (last five minutes): NaN
      Maximum tombstones per slice (last five minutes): 0
      Dropped Mutations: 1 bytes
```


This output tells us the following statistics about the Cassandra cluster health:

- **SSTable count** - This will tell how many SSTables containing data for this table. High value of SSTable (~ more than 10) indicate that compaction is not happening regularly.
- **Write count** - Constant increase in this value indicates there are continuous writes happening on this table. This is a good indicator that migration utility is actually continuously writing to Cassandra.
- **Percent repaired** - Consistency repair is actually an important mechanism for making sure copies of data are shipped around the cluster to meet your specified replication factor. If it is less than 50%, then it means that 50% of data is not replicated properly and hence we may end up losing the data.
- **Dropped Mutations** - The number of mutations (INSERTs, UPDATEs or DELETEs) started on this table but not completed. A large number of dropped mutation means that node is overloaded.

Check write latency of table atokens:

```
[root@cass-set-1-0 builder]# nodetool tablehistograms oauth2 atokens
oauth2/atokens histograms
```

Percentile	SSTables	Write Latency (micros)	Read Latency (micros)	Partition Size (bytes)	Cell Count
50%	0.00	73.46	0.00	179	10
75%	0.00	126.93	0.00	179	10
95%	0.00	219.34	0.00	179	10
98%	0.00	263.21	0.00	179	10
99%	0.00	315.85	0.00	179	10
Min	0.00	29.52	0.00	150	9
Max	0.00	1386179.89	0.00	179	10

This output tells us the following statistics about the Cassandra cluster health:

- **Write latency** - In above example output (taken from cloud setup), the write latency has been maxed out to 1.38 seconds which is marginally lesser than configured value of `write_request_timeout_in_ms = 2 seconds` (/etc/cassandra/conf/cassandra.yml). if it gets higher than configured value then we may see `WriteTimeoutException` issue while writing records to Cassandra.

Check overall node status and heap memory:

```
[root@cass-set-1-0 builder]# nodetool info
ID : 7d134170-9dea-419f-8cc0-9f5692245269
```



```

Gossip active      : true
Thrift active      : false
Native Transport active: true
Load               : 3.04 MiB
Generation No      : 1554892036
Uptime (seconds)   : 170138
Heap Memory (MB)   : 425.22 / 460.81
Off Heap Memory (MB) : 0.08
Data Center        : dc2
Rack               : rack1
Exceptions         : 0
Key Cache          : entries 68, size 5.55 KiB, capacity 23 MiB, 82400 hits, 82665 requests, 0.997
recent hit rate, 14400 save period in seconds
Row Cache          : entries 0, size 0 bytes, capacity 0 bytes, 0 hits, 0 requests, NaN recent hit rate,
0 save period in seconds
Counter Cache      : entries 0, size 0 bytes, capacity 11 MiB, 0 hits, 0 requests, NaN recent hit rate,
7200 save period in seconds
Chunk Cache        : entries 33, size 2.06 MiB, capacity 83 MiB, 572 misses, 1288690 requests,
1.000 recent hit rate, NaN microseconds miss latency
Percent Repaired   : 0.0%
Token              : (invoke with -T/--tokens to see all 32 tokens)
[root@cass-set-1-0 builder]#

```

This output tells us the following statistics about the Cassandra cluster health:

- **Heap memory** - Too little heap memory availability out of total heap memory may cause slow operations. In this case, check with `compactionstats` and `tablestats` command to determine the root cause.

Check if there are pending tasks piling up:

```

[root@cass-set-0-0 builder]# nodetool compactionstats
pending tasks: 0

```

This output tells the following statistics about the Cassandra cluster health:

- It indicates the estimation of the work to be done. If it continues to increase then it indicates that the compaction strategy is not proper, or you are low on disk space.

Monitoring Local Edition Components

Local Edition is made up of several components - [NoSQL](#), [SQL](#), [Cache](#), [Log](#), [Traffic Manager](#), Configuration Data API, and [Cluster Manager](#). To be confident that Local Edition is healthy and serving traffic optimally, it is vital to monitor the health of the overall system, as well as the individual components. Health monitoring is achieved by ensuring the components are:

- alive
- reachable
- serving requests successfully in optimal time

Format of Metrics

Metrics are collected in JSON form in the following format:

```
{
  "tag": "tml-<containertype>.<hostname>.metrics.<metric type name>.<subtypes>.<...>",
  "ingestion_time": "<UTC time in ISO8601 form ex. 2019-03-28T13:35:28+00:00>",
  "<metric name>": "<String metric value>",
  "<metric name>": "<Number metric value>"
}
```

Resource Monitoring

Resource monitoring includes monitoring resources like CPU, memory, disk and Network I/O.

Container Metrics

These metrics denote the utilization of CPU, memory, network and disks assigned to a container or pod.

i Note: CPU, Memory and Network

In the absence of limits and reservations on the containers or pods, all containers and pods can utilize available resources of the node on which they are deployed and running.

Container CPU Metrics

Captured metrics reflect the percentage of CPU utilized by container/pod, user space and a distill of the usage per core.

CPU Metrics example:

```
{
  "time": 1554194126,
  "message": {
    "cpu_p": 41.88333333333333,
    "user_p": 25.033333333333335,
    "system_p": 16.85,
    "cpu0.p_cpu": 41.88333333333333,
    "cpu0.p_user": 25.033333333333335,
    "cpu0.p_system": 16.85,
    "ingestion_time": "2019-04-02T08:35:26+00:00",
    "tag": "tml-nosql.6c680d874676.metrics.cpu"
  }
}
```

Metric Name	Field Name	Units	Data Type	Notes
Total CPU consumption	cpu_p	%	number	Total CPU usage across all cores assigned to the container - includes user and Kernel processes if there are 4 cores the container can use, the percent usage can go up to 400%
CPU consumption by user processes	user_p	%	number	Total CPU used by user processes across all cores

Metric Name	Field Name	Units	Data Type	Notes
CPU consumption by kernel processes	system_p	%	number	Total CPU used by kernel processes across all cores.
Total usage per core N	cpuN	%	number	Usage of Core N by user and kernel processes
User processes usage of Core N	cpuN	%	number	Usage of Core N by user processes
Kernel processes usage of core N	cpuN	%	number	Usage of Core N by kernel processes.

Container Memory

Pod/Container memory metrics example:

```
{
  "time": 1554198840,
  "message": {
    "Mem.total": 4045520,
    "Mem.used": 3932664,
    "Mem.free": 112856,
    "Swap.total": 1928204,
    "Swap.used": 1483436,
    "Swap.free": 444768,
    "ingestion_time": "2019-04-02T09:54:00+00:00",
    "tag": "tml-log.6a6873b34d5e.metrics.mem"
  }
}
```

Metric Name	Field Name	Units	Data Type	Notes
Total memory (RAM)	Mem.total	bytes	Number	Total memory available to container or pod in bytes
Used memory (RAM)	Mem.used	bytes	Number	Memory utilized by container in bytes
Free memory (RAM)	Mem.free	bytes	Number	available free RAM in bytes
Total swap space	Swap.total	bytes	Number	Total swap space
Used swap space	Swap.used	bytes	Number	Used swap space
Free swap space	Swap.free	bytes	Number	Free swap space

Container Disk

Captured metrics reflects number of bytes read and written at the point in time.

Pod/ Container disk metrics example:

```
{
  "time": 1554193560,
  "message": {
    "read_size": 7029587968,
    "write_size": 14102749184,
    "ingestion_time": "2019-04-02T08:26:00+00:00",
    "tag": "tml-log.6a6873b34d5e.metrics.disk"
  }
}
```

Metric Name	Field Name	Units	Data Type
Total bytes read from disk	read_Size	bytes	Number
Total bytes written to disk	write_size	bytes	Number

Container Network

The network metrics are available per network interface like eth1, lo etc. The metrics captured reflect the transmit and receive size at the point in time.

Pod/Container Network metrics example:

```
{
  "time": 1554199020,
  "message": {
    "eth0.rx.bytes": 516319,
    "eth0.rx.packets": 1062,
    "eth0.rx.errors": 0,
    "eth0.tx.bytes": 61578,
    "eth0.tx.packets": 893,
    "eth0.tx.errors": 0,
    "ingestion_time": "2019-04-02T09:57:00+00:00",
    "tag": "tml-log.6a6873b34d5e.metrics.netif"
  }
}
```

Metric Name	Field Name	Units	Data Type	Notes
Bytes transmitted on a netif_name	netif_name	bytes	Number	Total bytes transmitted for the particular network interface.
Packets transmitted on a netif_name	netif_name	Packet	Number	Total packets transmitted for the particular network interface.
Errors in transmitting packets on a netif_name	netif_name	Packet	Number	Number of packets failed to be transmitted for particular network interface due to window, carrier, aborted, or heartbeat errors.
Bytes received on a netif_name	netif_name	bytes	Number	Total bytes received for the particular network interface.
Packets received on a netif_name	netif_name	Packet	Number	Total packets received for the particular network interface.
Errors receiving	netif_	Packet	Number	Number of packets dropped.

Metric Name	Field Name	Units	Data Type	Notes
packets on a netif_name	name			

Common Process Metrics

```
{
  "time": 1554199440,
  "message": {
    "alive": true,
    "proc_name": "td-agent-bit",
    "pid": 2156,
    "mem.VmPeak": 83856000,
    "mem.VmSize": 83852000,
    "mem.VmLck": 0,
    "mem.VmHWM": 7416000,
    "mem.VmRSS": 3412000,
    "mem.VmData": 31028000,
    "mem.VmStk": 132000,
    "mem.VmExe": 4184000,
    "mem.VmLib": 5352000,
    "mem.VmPTE": 140000,
    "mem.VmSwap": 2040000,
    "fd": 65,
    "cpu.usage": 0.1,
    "mem.usage": 0.1,
    "uptime": 1478563,
    "LOG_FORMAT_VERSION": "2.0",
    "ingestion_time": "2019-04-02T10:04:00+00:00",
    "tag": "tml-log.6a6873b34d5e.metrics.proc.td-agent-bit"
  }
}
```

Metric Name	Field Name	Unit	Data Type	Notes
Process status	alive		Boolean	The status of the process

Metric Name	Field Name	Unit	Data Type	Notes
Process name	proc_name		String	Name of the process as identified by /proc/pid/cmd
Peak virtual memory usage	mem.VmPeak	bytes	Number	Max memory used by this process so far
Virtual memory size	mem.VmSize	bytes	Number	
Current mlocked memory	mem.VmLck	bytes	Number	Amount of memory locked by the process. This memory is released after the process exits.
Peak RAM used	mem.VmHWM	bytes	Number	
Current RAM being used	mem.VmRSS	bytes	Number	
Size of data	mem.VmData	bytes	Number	
Size of stack	mem.VmStk	bytes	Number	
Size of text segment	mem.VmExe	bytes	Number	
Shared library mem usage	mem.VmLib	bytes	Number	
Current swap space used	mem.VmSwap	bytes	Number	

Metric Name	Field Name	Unit	Data Type	Notes
CPU Usage	cpu.usage	%	Double	Gives the percentage of CPU used by the process
Memory usage	mem.usage	%	Double	Gives the percentage of memory used by the process
Age of process	uptime	seconds	Number	Gives the age of the process
Version	LOG_FORMAT_VERSION		String	Current version of format of the logs

Process List

Processes on all containers

Process Name	Description
Containeragent	The Local Edition container agent which manages all processes running inside a container.
td-agent-bit	The Log and metrics forwarder. It forwards all logs to the Log service.

Per Container Processes

Container Name	Process Name	Description
TM	proxy	Traffic Manager Proxy (embedded jetty)
Sql	Jetty	On-Prem Loader - syncs with MOM in tethered mode
Sql	mysqld	Service for MySQL
NoSql (seed	Cassandra	

Container Name	Process Name	Description
and non-seed)		
NoSql (only on non-seed)	Jetty	Jetty server hosting the API Management - Local Edition Registry Java webapp
Cache	Memcached	Six processes, one each for pools 11211, 11212, 11213, 1124, 11215 and, 11216
Cache	pxrt	The memcached loader, which keeps memcached up-to-date with changes to service definitions, packages et al.
Api	lighthttpd	CGI server supporting PHP CGI
API	memcached	Two processes, one each for pools 11211 and 11214
API	pxrt	Embedded Jetty server hosting the V3 API
API	php-cgi	~20 php-cgi processes - workers which execute a V2 API request
CM	Jetty	Jetty server hosting the certificate manager Java webapp.
logservice	td-agent	Log collector and forwarder. Gets logs from other containers and forwards them to user-chosen destination. 1 supervisor + 9 Workers
logservice	java	Process that syncs access logs to Boomi Cloud API Management in "tethered" mode.

Diagnostic Recipe / Alerts

Metric	Field Name / Computation	Notes
Is Process Alive	alive	The process status metrics are captured every minute Low water mark is first time if `alive=false` and High watermark If it continues for next 5 minutes, for example, `alive=false` for next 5 times the process metrics are gathered.
Continuous high memory usage	mem.VmHWM / ? > .8	Process / Expected usage / Water mark • memcached 11214 / This memcached pool will take up more memory than the other pools. / low water mark • td-agent-bit / The memory utilization by this should be in the order of MBs. / low water mark • Traffic Manager (javaproxy) / The memory utilization by the Traffic Manager will see spikes and troughs but the average utilization will be mid-range to low water mark. / low water mark • MySqlD / TBD • Cassandra / During replication cycles, Cassandra will utilize higher, for example > low water mark

Component Metrics Monitoring

The health of the component can be determined by the percentage of free resources that are available to the component. For each component, the resource definition could be different.

Component Type	Resources to monitor
Traffic Manager	Available threads in a pool, available heap, etc.
SQL	Available connections, available InnoDB buffer pool, available QCache, etc.
Cache	Available connections, connection throttling, available memory, Eviction rate, Hit Rate, etc.
NoSQL	Active and Pending Tasks, Dropped Messages, Read and Write Latencies, Replication Lags, Available Heap, Available Disk, etc.
Log	Available threads to process API requests, available heap size, Garbage Collection time, etc.

If the component metrics are out of normal range, the components can be tuned to perform optimally, and also be scaled horizontally or vertically. Horizontal scaling means adding more instances of components (except for SQL); vertical scaling means adding more resources to existing components to serve optimally.

Traffic Manager Metrics

Jetty Connection/Request Thread Pool Metrics:

```
{
  "time": 1554204598,
  "message": {
    "0.connectionData.0.AvgConnectionDuration": 36.404666666666664,
    "0.connectionData.0.AvgRequestsPerConnection": 1,
    "0.connectionData.0.ConnectionDurationStdDev": 0.031923032437411084,
    "0.connectionData.0.Connections": 1,
    "0.connectionData.0.ConnectorUptime": "0hours2minutes7seconds",
    "0.connectionData.0.ConnectorUptimeSec": 127,
    "0.connectionData.0.Keystore": "NonSSL",
    "0.connectionData.0.MaxConnectionDuration": 36.451,
    "0.connectionData.0.MaxConnections": 2,
    "0.connectionData.0.MaxRequestsPerConnection": 1,
    "0.connectionData.0.Port": 80,
```

```

"0.connectionData.0.RequestsPerConnectionStdDev": 0,
"0.connectionData.0.TotalConnectionDuration": "0hours1minute49seconds",
"0.connectionData.0.TotalConnectionDurationSec": 109,
"0.connectionData.0.TotalConnections": 4,
"0.connectionData.0.TotalRequests": 3,
"0.connectionData.1.AvgConnectionDuration": "NaN",
"0.connectionData.1.AvgRequestsPerConnection": "NaN",
"0.connectionData.1.ConnectionDurationStdDev": 0,
"0.connectionData.1.Connections": 0,
"0.connectionData.1.ConnectorUptime": "0hours2minutes7seconds",
"0.connectionData.1.ConnectorUptimeSec": 127,
"0.connectionData.1.Keystore": "/etc/mashery-server-ssl/mashery-local.jks",
"0.connectionData.1.MaxConnectionDuration": 0,
"0.connectionData.1.MaxConnections": 0,
"0.connectionData.1.MaxRequestsPerConnection": 0,
"0.connectionData.1.Port": 443,
"0.connectionData.1.RequestsPerConnectionStdDev": 0,
"0.connectionData.1.TotalConnectionDuration": "0hours0minutes0seconds",
"0.connectionData.1.TotalConnectionDurationSec": 0,
"0.connectionData.1.TotalConnections": 0,
"0.connectionData.1.TotalRequests": 0,
"0.threadPoolIdleThreads": 46,
"0.threadPoolMaxThreads": 500,
"0.threadPoolMinThreads": 50,
"0.threadPoolThreads": 50,
"ingestion_time": "2019-04-02T11:29:58+00:00",
"tag": "tml-tm.4a193af10aba.metrics.jym.proxy.jetty_connection_pool"
}
}

```

**Note:**

- 0.connectionData.0 is the thread pool for plain HTTP requests (port 80).
- 0.connectionData.1 is the thread pool for HTTPS requests (port 443).

Name	Field	Notes	Details
Max requests threads	0.threadPoolMaxThreads	Denotes the maximum concurrent requests	

Name	Field	Notes	Details
		possible on this Traffic manager instance	
Initial number of threads	0.threadPoolMinThreads	Threads to be created on startup	
Current created threads	0.threadPoolThreads	Number of threads created in the pool	
Idle Threads	0.threadPoolIdleThreads	Number of threads created but not utilized	A high value of idle threads denotes low utilization of this instance. Check if other traffic manager instances are over-loaded
Total connections since start	0.connectionData.0.TotalConnections	Number of connection requests to this instance	
Avg time for requests	0.connectionData.0.AvgConnectionDuration	Average time taken by the this instance to server traffic	A low value denotes high throughput

Diagnostic Recipe / Alerts

Metric	Field Name / Computation	Notes
Efficiency	ratio of 0.threadPoolThreads/0.threadPoolMaxThreads	This ratio provides the utilization of this Traffic Manager. An ideal range is 75%-80%. Low water mark - 80% High water mark 90% A continuous breach of high water mark indicates that this Traffic Manager instance is over loaded.
Under-utilization	ratio of 0.threadPoolIdleThreads/0.threadPoolMaxThreads	A very high ratio denotes that the

Metric	Field Name / Computation	Notes
		Traffic Manager is handling fewer calls and might indicate deficiency in load balancing rules.
Max connections	0.connectionData.0.MaxConnections+0.connectionData.0.MaxConnections	This is the maximum limit this Traffic Manager instance can serve requests.

Memory Utilization

Memory utilization is a combination of current usage + garbage collection (GC) metrics.

Current Usage

Current Usage: Heap

```
{
  "time": 1554205856,
  "message": {
    "0.HeapMemoryUsage.committed": 39088128,
    "0.HeapMemoryUsage.init": 33554432,
    "0.HeapMemoryUsage.max": 1037959168,
    "0.HeapMemoryUsage.used": 25810688,
    "ingestion_time": "2019-04-02T11:50:56+00:00",
    "tag": "tml-tm.4a193af10aba.metrics.jvm.proxy.memory.heap"
  }
}
```

Current Usage: Non Heap


```
{
  "time": 1554205856,
  "message": {
    "0.NonHeapMemoryUsage.committed": 58982400,
    "0.NonHeapMemoryUsage.init": 2555904,
    "0.NonHeapMemoryUsage.max": -1,
    "0.NonHeapMemoryUsage.used": 54043656,
    "ingestion_time": "2019-04-02T11:50:56+00:00",
    "tag": "tml-tm.4a193af10aba.metrics.jvm.proxy.memory.non_heap"
  }
}
```

Diagnostic Recipe / Alerts

Metric	Field Name / Computation	Unit	Data Type	Notes	Details
Current usage	0.HeapMemoryUsage.used + 0.NonHeapMemoryUsage.used	Bytes	Number		https://docs.oracle.com/javase/9/docs/api/java/lang/management/MemoryUsage.html
Committed memory	0.NonHeapMemoryUsage.committed + 0.HeapMemoryUsage.committed	Bytes	Number	Guaranteed to be available to the JVM	
Maximum memory	Mostly undefined to rely on available resources of system	Bytes	Number	Total allocated memory to the JVM	

Metric	Field Name / Computation	Unit	Data Type	Notes	Details
High Usage	Ratio of Used/Committed	N/A	Number	Used should always be lower than Committed. If Used > Committed, then the JVM can run Out of memory and stop. A high watermark of 80% can indicate that the JVM needs more memory	

Garbage Collection

Garbage collection (GC) for the JVM is split into Young and Old GC. Old GC should be low for Traffic manager.

Old GC can also include full GC and sometimes only tenured space reclamation.

Young GC tries to recover transient, short-lived objects from Eden and Survivor space. For more details, see https://docs.oracle.com/javase/8/docs/technotes/guides/management/jconsole.html#gd_eao.

For traffic managers, the tenured space should reflect low usage vis-a-vis young space. The GC from the Young space should reflect greater collection than the tenured.

i Note: The Young GC captures values for **Tenured** space and the Old GC captures values for **Eden** space. These can be safely ignored since they denote movement of objects between the two spaces during the life of the JVM.

Garbage Collection: OLD GC

```
{
  "time": 1554206878,
  "message": {
    "memoryUsageAfterGc.CodeCache.committed": 9175040,
    "memoryUsageAfterGc.CodeCache.init": 2555904,
    "memoryUsageAfterGc.CodeCache.max": 251658240,
    "memoryUsageAfterGc.CodeCache.used": 9141504,
    "memoryUsageAfterGc.CompressedClassSpace.committed": 4849664,
    "memoryUsageAfterGc.CompressedClassSpace.init": 0,
    "memoryUsageAfterGc.CompressedClassSpace.max": 1073741824,
    "memoryUsageAfterGc.CompressedClassSpace.used": 3764240,
    "memoryUsageAfterGc.EdenSpace.committed": 9109504,
    "memoryUsageAfterGc.EdenSpace.init": 8912896,
    "memoryUsageAfterGc.EdenSpace.max": 286326784,
    "memoryUsageAfterGc.EdenSpace.used": 0,
    "memoryUsageAfterGc.Metaspace.committed": 36306944,
    "memoryUsageAfterGc.Metaspace.init": 0,
    "memoryUsageAfterGc.Metaspace.max": -1,
    "memoryUsageAfterGc.Metaspace.used": 32641584,
    "memoryUsageAfterGc.SurvivorSpace.committed": 1114112,
    "memoryUsageAfterGc.SurvivorSpace.init": 1114112,
    "memoryUsageAfterGc.SurvivorSpace.max": 35782656,
    "memoryUsageAfterGc.SurvivorSpace.used": 0,
    "memoryUsageAfterGc.TenuredGen.committed": 22413312,
    "memoryUsageAfterGc.TenuredGen.init": 22413312,
    "memoryUsageAfterGc.TenuredGen.max": 715849728,
    "memoryUsageAfterGc.TenuredGen.used": 16019432,
    "ingestion_time": "2019-04-02T12:07:58+00:00",
    "tag": "tml-tm.4a193af10aba.metrics.jvm.proxy.old_gc"
```

```

    }
  }
  "time": 1554206878,
  "message": {
    "memoryUsageBeforeGc.CodeCache.committed": 9175040,
    "memoryUsageBeforeGc.CodeCache.init": 2555904,
    "memoryUsageBeforeGc.CodeCache.max": 251658240,
    "memoryUsageBeforeGc.CodeCache.used": 9141504,
    "memoryUsageBeforeGc.CompressedClassSpace.committed": 4849664,
    "memoryUsageBeforeGc.CompressedClassSpace.init": 0,
    "memoryUsageBeforeGc.CompressedClassSpace.max": 1073741824,
    "memoryUsageBeforeGc.CompressedClassSpace.used": 3764240,
    "memoryUsageBeforeGc.EdenSpace.committed": 9109504,
    "memoryUsageBeforeGc.EdenSpace.init": 8912896,
    "memoryUsageBeforeGc.EdenSpace.max": 286326784,
    "memoryUsageBeforeGc.EdenSpace.used": 7199984,
    "memoryUsageBeforeGc.Metaspace.committed": 36306944,
    "memoryUsageBeforeGc.Metaspace.init": 0,
    "memoryUsageBeforeGc.Metaspace.max": -1,
    "memoryUsageBeforeGc.Metaspace.used": 32641584,
    "memoryUsageBeforeGc.SurvivorSpace.committed": 1114112,
    "memoryUsageBeforeGc.SurvivorSpace.init": 1114112,
    "memoryUsageBeforeGc.SurvivorSpace.max": 35782656,
    "memoryUsageBeforeGc.SurvivorSpace.used": 1114112,
    "memoryUsageBeforeGc.TenuredGen.committed": 22413312,
    "memoryUsageBeforeGc.TenuredGen.init": 22413312,
    "memoryUsageBeforeGc.TenuredGen.max": 715849728,
    "memoryUsageBeforeGc.TenuredGen.used": 15496552,
    "ingestion_time": "2019-04-02T12:07:58+00:00",
    "tag": "tml-tm.4a193af10aba.metrics.jvm.proxy.old_gc"
  }
}
  "time": 1554206878,
  "message": {
    "collectionCount": 2,
    "collectionTime": 88,
    "name": "MarkSweepCompact",
    "gcStartTime": 4300,
    "gcEndTime": 4353,
    "gcDuration": 53,
    "ingestion_time": "2019-04-02T12:07:58+00:00",
    "tag": "tml-tm.4a193af10aba.metrics.jvm.proxy.old_gc"
  }
}

```

Garbage Collection: Young GC

```
{
  "time": 1554207478,
  "message": {
    "memoryUsageAfterGc.CodeCache.committed": 15925248,
    "memoryUsageAfterGc.CodeCache.init": 2555904,
    "memoryUsageAfterGc.CodeCache.max": 251658240,
    "memoryUsageAfterGc.CodeCache.used": 15751744,
    "memoryUsageAfterGc.CompressedClassSpace.committed": 5242880,
    "memoryUsageAfterGc.CompressedClassSpace.init": 0,
    "memoryUsageAfterGc.CompressedClassSpace.max": 1073741824,
    "memoryUsageAfterGc.CompressedClassSpace.used": 4136776,
    "memoryUsageAfterGc.EdenSpace.committed": 11010048,
    "memoryUsageAfterGc.EdenSpace.init": 8912896,
    "memoryUsageAfterGc.EdenSpace.max": 286326784,
    "memoryUsageAfterGc.EdenSpace.used": 0,
    "memoryUsageAfterGc.Metaspace.committed": 41418752,
    "memoryUsageAfterGc.Metaspace.init": 0,
    "memoryUsageAfterGc.Metaspace.max": -1,
    "memoryUsageAfterGc.Metaspace.used": 37438280,
    "memoryUsageAfterGc.SurvivorSpace.committed": 1376256,
    "memoryUsageAfterGc.SurvivorSpace.init": 1114112,
    "memoryUsageAfterGc.SurvivorSpace.max": 35782656,
    "memoryUsageAfterGc.SurvivorSpace.used": 547192,
    "memoryUsageAfterGc.TenuredGen.committed": 26701824,
    "memoryUsageAfterGc.TenuredGen.init": 22413312,
    "memoryUsageAfterGc.TenuredGen.max": 715849728,
    "memoryUsageAfterGc.TenuredGen.used": 18922936,
    "ingestion_time": "2019-04-02T12:17:58+00:00",
    "tag": "tml-tm.4a193af10aba.metrics.jvm.proxy.young_gc"
  }
}
{
  "time": 1554207478,
  "message": {
    "memoryUsageBeforeGc.CodeCache.committed": 15925248,
    "memoryUsageBeforeGc.CodeCache.init": 2555904,
    "memoryUsageBeforeGc.CodeCache.max": 251658240,
    "memoryUsageBeforeGc.CodeCache.used": 15751744,
    "memoryUsageBeforeGc.CompressedClassSpace.committed": 5242880,
    "memoryUsageBeforeGc.CompressedClassSpace.init": 0,
    "memoryUsageBeforeGc.CompressedClassSpace.max": 1073741824,
    "memoryUsageBeforeGc.CompressedClassSpace.used": 4136776,
    "memoryUsageBeforeGc.EdenSpace.committed": 11010048,
    "memoryUsageBeforeGc.EdenSpace.init": 8912896,
    "memoryUsageBeforeGc.EdenSpace.max": 286326784,
```

```

"memoryUsageBeforeGc.EdenSpace.used": 11010048,
"memoryUsageBeforeGc.Metaspace.committed": 41418752,
"memoryUsageBeforeGc.Metaspace.init": 0,
"memoryUsageBeforeGc.Metaspace.max": -1,
"memoryUsageBeforeGc.Metaspace.used": 37438280,
"memoryUsageBeforeGc.SurvivorSpace.committed": 1376256,
"memoryUsageBeforeGc.SurvivorSpace.init": 1114112,
"memoryUsageBeforeGc.SurvivorSpace.max": 35782656,
"memoryUsageBeforeGc.SurvivorSpace.used": 523144,
"memoryUsageBeforeGc.TenuredGen.committed": 26701824,
"memoryUsageBeforeGc.TenuredGen.init": 22413312,
"memoryUsageBeforeGc.TenuredGen.max": 715849728,
"memoryUsageBeforeGc.TenuredGen.used": 18922936,
"ingestion_time": "2019-04-02T12:17:58+00:00",
"tag": "tml-tm.4a193af10aba.metrics.jvm.proxy.young_gc"
}
}{
"time": 1554207478,
"message": {
"collectionCount": 19,
"collectionTime": 83,
"name": "Copy",
"gcStartTime": 2950514,
"gcEndTime": 2950518,
"gcDuration": 4,
"ingestion_time": "2019-04-02T12:17:58+00:00",
"tag": "tml-tm.4a193af10aba.metrics.jvm.proxy.young_gc"
}
}
}

```

Diagnostic Recipes / Alerts

Oracle's Garbage Collection Tuning guide:

<https://docs.oracle.com/javase/8/docs/technotes/guides/vm/gctuning/index.html>.

JVM Threads

```

{
"time": 1555422289,
"message": {
"tag": "tml-tm.tm-deploy-0-5d6f55b566-zdplp.metrics.jvm.proxy.threading",
"ingestion_time": "2019-04-16T13:44:49+00:00",
"TML_CLUSTER_NAME": "Tibco Mashery Local Reference Cluster Pune",
"TML_ZONE_NAME": "us-east-1e",

```

```
"0.AllThreadIds.0": 151,  
"0.AllThreadIds.1": 150,  
"0.AllThreadIds.2": 149,  
"0.AllThreadIds.3": 105,  
"0.AllThreadIds.4": 104,  
"0.AllThreadIds.5": 100,  
"0.AllThreadIds.6": 99,  
"0.AllThreadIds.7": 98,  
"0.AllThreadIds.8": 97,  
"0.AllThreadIds.9": 96,  
"0.AllThreadIds.10": 95,  
"0.AllThreadIds.11": 94,  
"0.AllThreadIds.12": 93,  
"0.AllThreadIds.13": 92,  
"0.AllThreadIds.14": 91,  
"0.AllThreadIds.15": 90,  
"0.AllThreadIds.16": 89,  
"0.AllThreadIds.17": 88,  
"0.AllThreadIds.18": 87,  
"0.AllThreadIds.19": 86,  
"0.AllThreadIds.20": 85,  
"0.AllThreadIds.21": 84,  
"0.AllThreadIds.22": 83,  
"0.AllThreadIds.23": 82,  
"0.AllThreadIds.24": 81,  
"0.AllThreadIds.25": 80,  
"0.AllThreadIds.26": 79,  
"0.AllThreadIds.27": 78,  
"0.AllThreadIds.28": 77,  
"0.AllThreadIds.29": 76,  
"0.AllThreadIds.30": 75,  
"0.AllThreadIds.31": 74,  
"0.AllThreadIds.32": 73,  
"0.AllThreadIds.33": 72,  
"0.AllThreadIds.34": 71,  
"0.AllThreadIds.35": 70,  
"0.AllThreadIds.36": 69,  
"0.AllThreadIds.37": 68,  
"0.AllThreadIds.38": 67,  
"0.AllThreadIds.39": 66,  
"0.AllThreadIds.40": 65,  
"0.AllThreadIds.41": 64,  
"0.AllThreadIds.42": 63,  
"0.AllThreadIds.43": 62,  
"0.AllThreadIds.44": 61,
```

```
"0.AllThreadIds.45": 60,  
"0.AllThreadIds.46": 59,  
"0.AllThreadIds.47": 58,  
"0.AllThreadIds.48": 57,  
"0.AllThreadIds.49": 56,  
"0.AllThreadIds.50": 55,  
"0.AllThreadIds.51": 54,  
"0.AllThreadIds.52": 53,  
"0.AllThreadIds.53": 52,  
"0.AllThreadIds.54": 51,  
"0.AllThreadIds.55": 50,  
"0.AllThreadIds.56": 49,  
"0.AllThreadIds.57": 48,  
"0.AllThreadIds.58": 47,  
"0.AllThreadIds.59": 46,  
"0.AllThreadIds.60": 45,  
"0.AllThreadIds.61": 44,  
"0.AllThreadIds.62": 43,  
"0.AllThreadIds.63": 42,  
"0.AllThreadIds.64": 41,  
"0.AllThreadIds.65": 40,  
"0.AllThreadIds.66": 39,  
"0.AllThreadIds.67": 38,  
"0.AllThreadIds.68": 37,  
"0.AllThreadIds.69": 36,  
"0.AllThreadIds.70": 35,  
"0.AllThreadIds.71": 34,  
"0.AllThreadIds.72": 33,  
"0.AllThreadIds.73": 32,  
"0.AllThreadIds.74": 31,  
"0.AllThreadIds.75": 30,  
"0.AllThreadIds.76": 29,  
"0.AllThreadIds.77": 28,  
"0.AllThreadIds.78": 27,  
"0.AllThreadIds.79": 26,  
"0.AllThreadIds.80": 25,  
"0.AllThreadIds.81": 24,  
"0.AllThreadIds.82": 21,  
"0.AllThreadIds.83": 20,  
"0.AllThreadIds.84": 19,  
"0.AllThreadIds.85": 17,  
"0.AllThreadIds.86": 15,  
"0.AllThreadIds.87": 12,  
"0.AllThreadIds.88": 11,  
"0.AllThreadIds.89": 10,
```



```

"0.AllThreadIds.90": 4,
"0.AllThreadIds.91": 3,
"0.AllThreadIds.92": 2,
"0.CurrentThreadCpuTime": 2345099757,
"0.CurrentThreadCpuTimeSupported": "true",
"0.CurrentThreadUserTime": 1760000000,
"0.DaemonThreadCount": 17,
"0.ObjectMonitorUsageSupported": "true",
"0.ObjectName.canonicalKeyPropertyListString": "type=Threading",
"0.ObjectName.canonicalName": "java.lang:type=Threading",
"0.ObjectName.domain": "java.lang",
"0.ObjectName.domainPattern": "false",
"0.ObjectName.keyPropertyList.type": "Threading",
"0.ObjectName.keyPropertyListString": "type=Threading",
"0.ObjectName.pattern": "false",
"0.ObjectName.propertyListPattern": "false",
"0.ObjectName.propertyPattern": "false",
"0.ObjectName.propertyValuePattern": "false",
"0.PeakThreadCount": 95,
"0.SynchronizerUsageSupported": "true",
"0.ThreadAllocatedMemoryEnabled": "true",
"0.ThreadAllocatedMemorySupported": "true",
"0.ThreadContentionMonitoringEnabled": "false",
"0.ThreadContentionMonitoringSupported": "true",
"0.ThreadCount": 93,
"0.ThreadCpuTimeEnabled": "true",
"0.ThreadCpuTimeSupported": "true",
"0.TotalStartedThreadCount": 144
}
}

```

Diagnostic Recipe / Alerts

Metric	Field Name / Computation	Unit	Data Type	Details
Max Used Threads	0.PeakThreadCount	NA	Number	Represents peak live thread count since the JVM started or the peak was reset.
Current Thread	0.ThreadCount	NA	Number	Represents current number of live

Metric	Field Name / Computation	Unit	Data Type	Details
Count				daemon and non-daemon threads.
Started Thread Count	0.TotalStartedThreadCount	NA	Number	Total number of threads created and/or started since the JVM started.

SQL Container Metrics (MySQL DB)

The following table describes the metrics that are required for monitoring the health of the MySQL DB services.



Note:

- Data field is the name of the metric in the captured data.
- Given fields are key fields which can be used to identify any abnormalities.
- For more information on MySQL server status monitoring, see <https://dev.mysql.com/doc/refman/8.0/en/server-status-variables.html>.

MySQL DB Stats Sample:

DB-Stats Sample

```
{
  "Connection_errors_accept": 0,
  "Connection_errors_internal": 0,
  "Connection_errors_max_connections": 0,
  "Connection_errors_peer_address": 0,
  "Connection_errors_select": 0,
  "Connection_errors_tcpwrap": 0,
  "Connections": 234,
  "Delayed_insert_threads": 0,
  "Innodb_buffer_pool_dump_status": "Dumping",
  "Innodb_buffer_pool_load_status": "Buffer",
  "Innodb_buffer_pool_resize_status": "",
  "Innodb_buffer_pool_pages_data": 1802,
  "Innodb_buffer_pool_bytes_data": 29523968,
```

```

"Innodb_buffer_pool_pages_dirty": 0,
"Innodb_buffer_pool_bytes_dirty": 0,
"Innodb_buffer_pool_pages_flushed": 2834,
"Innodb_buffer_pool_pages_free": 6302,
"Innodb_buffer_pool_pages_misc": 87,
"Innodb_buffer_pool_pages_total": 8191,
"Innodb_buffer_pool_read_ahead_rnd": 0,
"Innodb_buffer_pool_read_ahead": 0,
"Innodb_buffer_pool_read_ahead_evicted": 0,
"Innodb_buffer_pool_read_requests": 3247529,
"Innodb_buffer_pool_reads": 1603,
"Innodb_buffer_pool_wait_free": 0,
"Innodb_buffer_pool_write_requests": 279100,
"Innodb_data_fsyncs": 11193,
"Innodb_data_pending_fsyncs": 0,
"Innodb_data_pending_reads": 0,
"Innodb_data_pending_writes": 0,
"Innodb_data_read": 29266432,
"Innodb_data_reads": 2006,
"Innodb_data_writes": 13727,
"Innodb_data_written": 126338560,
"Innodb_dblwr_pages_written": 2799,
"Innodb_dblwr_writes": 148,
"Innodb_log_waits": 0,
"Innodb_log_write_requests": 36527,
"Innodb_log_writes": 10540,
"Innodb_os_log_fsyncs": 10722,
"Innodb_os_log_pending_fsyncs": 0,
"Innodb_os_log_pending_writes": 0,
"Innodb_os_log_written": 33955840,
"Innodb_page_size": 16384,
"Innodb_pages_created": 134,
"Innodb_pages_read": 1668,
"Innodb_pages_written": 2834,
"Innodb_row_lock_current_waits": 0,
"Innodb_row_lock_time": 0,
"Innodb_row_lock_time_avg": 0,
"Innodb_row_lock_time_max": 0,
"Innodb_row_lock_waits": 0,
"Innodb_rows_deleted": 11712,
"Innodb_rows_inserted": 11924,
"Innodb_rows_read": 4709484,
"Innodb_rows_updated": 90,
"Innodb_num_open_files": 76,
"Innodb_truncated_status_writes": 0,

```

```

"Innodb_available_undo_logs": 128,
"Max_used_connections": 6,
"Max_used_connections_time": "2019-03-25",
"Performance_schema_accounts_lost": 0,
"Performance_schema_cond_classes_lost": 0,
"Performance_schema_cond_instances_lost": 0,
"Performance_schema_digest_lost": 0,
"Performance_schema_file_classes_lost": 0,
"Performance_schema_file_handles_lost": 0,
"Performance_schema_file_instances_lost": 0,
"Performance_schema_hosts_lost": 0,
"Performance_schema_index_stat_lost": 0,
"Performance_schema_locker_lost": 0,
"Performance_schema_memory_classes_lost": 0,
"Performance_schema_metadata_lock_lost": 0,
"Performance_schema_mutex_classes_lost": 0,
"Performance_schema_mutex_instances_lost": 0,
"Performance_schema_nested_statement_lost": 0,
"Performance_schema_prepared_statements_lost": 0,
"Performance_schema_program_lost": 0,
"Performance_schema_rwlock_classes_lost": 0,
"Performance_schema_rwlock_instances_lost": 0,
"Performance_schema_session_connect_attrs_lost": 0,
"Performance_schema_socket_classes_lost": 0,
"Performance_schema_socket_instances_lost": 0,
"Performance_schema_stage_classes_lost": 0,
"Performance_schema_statement_classes_lost": 0,
"Performance_schema_table_handles_lost": 0,
"Performance_schema_table_instances_lost": 0,
"Performance_schema_table_lock_stat_lost": 0,
"Performance_schema_thread_classes_lost": 0,
"Performance_schema_thread_instances_lost": 0,
"Performance_schema_users_lost": 0,
"Qcache_queries_in_cache": 0,
"Queries": 415232,
"Slow_launch_threads": 0,
"Slow_queries": 0,
"Threads_cached": 2,
"Threads_connected": 4,
"Threads_created": 6,
"Threads_running": 1,
"Uptime": 19807,
"Uptime_since_flush_status": 19807,
"ingestion_time": "2019-03-25T15:07:38+00:00"
}

```

Name	Data Field	Unit	Data Type	Details
Connection_errors_accept	Connection_errors_accept	N/A	Number	The number of errors that occurred during calls to accept() on the listening port.
Connection_errors_max_connections	Connection_errors_max_connections	N/A	Number	The number of connections refused because the server max_connections limit was reached.
Connection_errors_peer_address	Connection_errors_peer_address	N/A	Number	The number of errors that occurred while searching for connecting client IP addresses.
Innodb_pages_read	Innodb_pages_read	N/A	Number	The number of pages read from the InnoDB buffer pool by operations on InnoDB tables.
Innodb_rows_deleted	Innodb_rows_deleted	N/A	Number	The number of rows deleted from InnoDB tables.
Innodb_rows_inserted	Innodb_rows_inserted	N/A	Number	The number of rows inserted into InnoDB tables.
Innodb_rows_read	Innodb_rows_read	N/A	Number	The number of rows read from InnoDB tables.
Innodb_rows_updated	Innodb_rows_updated	N/A	Number	The number of rows updated in InnoDB tables.

Name	Data Field	Unit	Data Type	Details
Max_used_connections	Max_used_connections	N/A	Number	The maximum number of connections that have been in use simultaneously since the server started.
Max_used_connection_s_time	Max_used_connection_s_time	N/A	Time	The time at which Max_used_connections reached its current value.
Queries	Queries	N/A	Number	The number of statements executed by the server. This variable includes statements executed within stored programs, unlike the Questions variable. It does not count COM_PING or COM_STATISTICS commands. The discussion at the beginning of this section indicates how to relate this statement-counting status variable to other such variables.
Slow_queries	Slow_queries	N/A	Number	The number of queries that have taken more than long_query_time seconds. This counter increments regardless of whether the slow query log is enabled. For information about that log, see https://dev.mysql.com/doc/refman/8.0/en/slow-query-log.html .
Threads_cached	Threads_cached	N/A	Number	The number of threads in the thread cache.
Threads_connected	Threads_connected	N/A	Number	The number of currently open connections.
Threads_created	Threads_created	N/A	Number	The number of threads created to handle connections. If Threads_created is big, you may want to

Name	Data Field	Unit	Data Type	Details
				increase the thread_cache_size value. The cache miss rate can be calculated as Threads_created/Connections.
Threads_running	Threads_running	N/A	Number	The number of threads that are not sleeping.
Uptime	Uptime	Seconds	Number	The number of seconds that the server has been up.
Uptime_since_flush_status	Uptime_since_flush_status	Seconds	Number	The number of seconds since the most recent FLUSH STATUS statement.

Diagnostic Recipe and Alerts for MySQL

Metrics	Computation
Buffer utilization	$\text{Innodb\_buffer\_pool\_bytes\_data} / (\text{Innodb\_page\_size} * \text{Innodb\_buffer\_pool\_pages\_total})$
Rejected connections	$\text{Max\_used\_connections} / \text{Max\_Connections}$
	$\text{Qcache\_free\_blocks} / \text{Qcache\_total\_blocks}$

Cache Container Metrics (Memcached)

The following table describes the metrics that are required for monitoring the health of the Memcached services.

Note:

- Data field is the name of the metric in the captured data.
- Slab stats are prefixed by `sn_`, where `n` is the slab number.
- For more information on memcached monitoring, see <https://github.com/memcached/memcached/wiki/ServerMaint>.

Memcached Per pool metrics

```
{
  "time":1554455492,
  "message":{
    "tag":"tml-cache.cache-set-0-0.metrics.stats.memcached.11211",
    "ingestion_time":"2019-04-05T09:11:32+00:00",
    "pool_name":"mapis",
    "pool":11211,
    "pid":343,
    "uptime":6912,
    "time":1554455487,
    "version":"1.5.7",
    "libevent":"2.0.21-stable",
    "pointer_size":64,
    "rusage_user":"0.714312",
    "rusage_system":"0.878758",
    "max_connections":10240,
    "curr_connections":5,
    "total_connections":462,
    "rejected_connections":0,
    "connection_structures":8,
    "reserved_fds":20,
    "cmd_get":23,
    "cmd_set":2071,
    "cmd_flush":0,
    "cmd_touch":0,
    "get_hits":22,
    "get_misses":1,
    "get_expired":0,
    "get_flushed":0,
    "delete_misses":0,
    "delete_hits":0,
    "incr_misses":0,
    "incr_hits":0,
    "decr_misses":0,
    "decr_hits":0,
    "cas_misses":0,
```



```
"cas_hits":0,  
"cas_badval":0,  
"touch_hits":0,  
"touch_misses":0,  
"auth_cmds":0,  
"auth_errors":0,  
"bytes_read":3015690,  
"bytes_written":843044,  
"limit_maxbytes":402653184,  
"accepting_conns":1,  
"listen_disabled_num":0,  
"time_in_listen_disabled_us":0,  
"threads":4,  
"conn_yields":0,  
"hash_power_level":16,  
"hash_bytes":524288,  
"hash_is_expanding":0,  
"slab_reassign_rescues":0,  
"slab_reassign_chunk_rescues":0,  
"slab_reassign_evictions_nomem":0,  
"slab_reassign_inline_reclaim":0,  
"slab_reassign_busy_items":0,  
"slab_reassign_busy_deletes":0,  
"slab_reassign_running":0,  
"slabs_moved":0,  
"lru_crawler_running":0,  
"lru_crawler_starts":3825,  
"lru_maintainer_juggles":34718,  
"malloc_fails":0,  
"log_worker_dropped":0,  
"log_worker_written":0,  
"log_watcher_skipped":0,  
"log_watcher_sent":0,  
"bytes":1545247,  
"curr_items":1015,  
"total_items":2071,  
"slab_global_page_pool":0,  
"expired_unfetched":0,  
"evicted_unfetched":0,  
"evicted_active":0,  
"evictions":0,  
"reclaimed":0,  
"crawler_reclaimed":0,  
"crawler_items_checked":12180,  
"lrutail_reflocked":0,
```

```

    "moves_to_cold":2071,
    "moves_to_warm":1,
    "moves_within_lru":20,
    "direct_reclaims":0,
    "lru_bumps_dropped":0
  }
}

```

Memcached Per pool per slab metrics

```

{
  "time":1554455492,
  "message":{
    "tag":"tml-cache.cache-set-0-0.metrics.stats.memcached.11211",
    "ingestion_time":"2019-04-05T09:11:32+00:00",
    "pool_name":"mapis",
    "pool":11211,
    "slab_class":1,
    "chunk_size":0,
    "chunk_per_page":0,
    "total_pages":0,
    "total_chunks":10922,
    "used_chunks":3,
    "free_chunks":10919,
    "free_chunks_end":0,
    "mem_requested":271,
    "get_hits":22,
    "cmd_set":47,
    "delete_hits":0,
    "incr_hits":0,
    "decr_hits":0,
    "cas_hits":0,
    "cas_badval":0,
    "touch_hits":0
  }
}

```

Name	Data Field	Unit	Data Type	Notes	Details
Pool Name	pool_name	N/A	String	Possible values counter,	The pool name maps to a Pool

Name	Data Field	Unit	Data Type	Notes	Details
				services, mapis, httpclientsecurity, packager, content	
pool	pool	N/A	Number	The port number on which the pool is listening for Memcached function calls Possible values 11211, 11212, 11213, 11214, 11215, 11216	11211 = mapis, 11212 = counter, 11213 = content, 11214 = services, 11215 = packages, 11216 = httpsclientsecurity
Particular slab	slab_class	N/A	Number	For each Memcache, there could be multiple slabs, slab_class denotes which slab it is.	
Process id	pid	N/A	Number	Process id of memcached services- there will be one PID per memcached pool	
Up time	uptime	N/A	Number	Time since start of process - denotes how long the process has been active.	

Name	Data Field	Unit	Data Type	Notes	Details
Maximum connections	max_connections	N/A	Number	Max connections allowed for this pool	
Total connections	total_connections	N/A	Number	Total number of successful connections since start	
Current client connections	curr_connections	N/A	Number	Number of Memcached clients connected at this point	
Total GETs	cmd_get	N/A	Number	Number of times a Get request was made- includes both successful and unsuccessful	
Total SETs	cmd_set	N/A	Number	Number of times a Set request was made - includes both successful and unsuccessful	
Total Get hits	get_hits	N/A	Number	Number of successful GET requests	
Total GET misses	get_misses	N/A	Number	Number of failed "get" requests because nothing was cached for	

Name	Data Field	Unit	Data Type	Notes	Details
				requested key.	
get hits	get_hits	N/A	Number	Number of hits for the slab - this should be > sn_cmd_set	
chunk size	chunk_size	Bytes	Number	Size of item chunks.	
Number of chunks in a page	chunks_per_page	Bytes	Number	Total number of chunks in a Page - This will compute as chunk_size*total_chunks = 1MB approximately.	

Diagnostic Recipe and Alerts for Memcached

On an ongoing basis (using ELK, Loglogic, Splunk, or other monitoring tools), you can keep track of the health of the Memcached pools.

Metric	Field Name / Computation	Notes
Hit rate	$\text{get\_hits}/(\text{get\_hits}+\text{get\_misses})$	The computed ratio should be nearly equal to 1 at that collection time.
Fill ratio/percent	$\text{bytes}/\text{limit\_maxbytes}$	Low water mark 85%; High water mark 90%.

NoSQL Container Metrics (Cassandra)

NoSQL Compaction Stats

1. Compaction Information

```
{
  "compaction.throughput": "16MB/s",
  "compaction.pending_task": "0"
}
```

Name	Data Field	Unit	Data Type	Notes	Details
Compaction Throughput	compaction.throughput	MB/Second	String	Default value is 16 MB/s	Defines the rate at which consolidated SSTables are written, and in turn helps to mitigate the tendency of small SSTables to accumulate during compaction. For more information, see https://docs.datastax.com/en/archived/cassandra/2.1/cassandra/operations/ops_configure_compaction_t.html .
Compaction Pending Tasks	compaction.pending_task	N/A	String	It should be 0.	Indicates the estimation of the work to be done. If it continues to increase, it indicates that the compaction strategy is not working, or disk space is low. For more information, see thelastpickle.com/blog/2017/03/16/compaction-nuance.html .

NoSql GC Stats

```
{
  "gc.interval": 301998,
  "gc.max_elapsed": 23,
  "gc.total_elapsed": 23,
  "gc.stddev_elapsed": 0,
  "gc.reclaimed": 83365208,
}
```

```
"gc.collections": 1,
"gc.memory_bytes": -1
}
```

Name	Data Field	Unit	Data Type	Details
GC Interval	gc.interval	Milliseconds	Number	Interval between last status check and current status check.
GC Max Time Elapsed	gc.max_elapsed	Milliseconds	Number	Maximum time taken by Garbage collection task in this interval.
GC Total Time Elapsed	gc.total_elapsed	Milliseconds	Number	Total time taken by all garbage collection tasks in this interval.
GC Stddev Elapsed	gc.stddev_elapsed	Milliseconds	Number	Standard deviation in Milliseconds.
GC Reclaimed Memory	gc.reclaimed	N/A	Number	Memory reclaimed in this interval.
GC Collection Count	gc.collections	Bytes	Number	Number of GC tasks performed in this interval.

NoSql Tokens KeySpace Info

```
{
  "time": {epoch_time},
  "message": {
    "tag": "tml-nosql.{HOSTNAME}.metrics.cassandra.keyspaceinfo",
    "ingestion_time": "{Time of ingestion to log service}",
    "oauth2.atokens.space.used.total": "0",
    "oauth2.atokens.space.used.live": "0",
    "oauth2.rtokens.space.used.total": "0",
    "oauth2.rtokens.space.used.live": "0",
  }
}
```

```
"oauth2.atokens.read.latency": "NaN",
"oauth2.atokens.write.latency": "NaN",
"oauth2.rtokens.read.latency": "NaN",
"oauth2.rtokens.write.latency": "NaN",
"oauth2.atokens.estimatedpartition.count": 0,
"oauth2.rtokens.estimatedpartition.count": 0,
"oauth2.atokens.dropped.mutation": "0",
"oauth2.rtokens.dropped.mutation": "0"
}
}
```

Name	Data Field	Unit	Data Type	Notes	Details
Oauth2.atokens key space	oauth2.atokens.space.used.total	Bytes	String		Represents the size of the column family (table) "atokens" in "Oauth2" key space including total number of bytes of disk space used by SSTables belonging to this table, including obsolete SSTables waiting to be GCd.

Name	Data Field	Unit	Data Type	Notes	Details
Oauth2.atokens key space	oauth2.atokens.space.used.live	Bytes	String		Represents the size of the column family (table) "atokens" in "Oauth2" key space having only total number of bytes of disk space used by all active SSTables belonging to this table.
Oauth2.rtokens key space	oauth2.rtokens.space.used.total	Bytes	String		Represents the size of the column family (table) "rtokens" in "Oauth2" key space including total number of bytes of disk space used by SSTables

Name	Data Field	Unit	Data Type	Notes	Details
					belonging to this table, including obsolete SSTables waiting to be GCd.
Oauth2.tokens key space	oauth2.tokens.space.used.live	Bytes	String		Represents the size of the column family (table) "rtokens" in "Oauth2" key space having only total number of bytes of disk space used by all active SSTables belonging to this table.
Oauth2.tokens Read Latency	oauth2.atokens.read.latency	Milliseconds	String		Round trip time in milliseconds to complete the most recent

Name	Data Field	Unit	Data Type	Notes	Details
					request to read the table
Oauth2.tokens Read Latency	oauth2.tokens.read.latency	Milliseconds	String		Round trip time in milliseconds to complete the most recent request to read the table
Oauth2.tokens Write Latency	oauth2.tokens.write.latency	Milliseconds	String		Round trip time in milliseconds to complete an update to the table. This should be minimal since a write need only be recorded in memory and appended to a durable commit log before it is

Name	Data Field	Unit	Data Type	Notes	Details
					acknowledged as a success.
Oauth2.tokens Write Latency	oauth2.tokens.write.latency	Milliseconds	String		Round trip time in milliseconds to complete an update to the table. This could vary depending upon frequent updates happening on the same data. This would delay read and may see read latency.
Oauth2.tokens Partition Count	oauth2.atokens.estimatedpartition.count	N/A	Number		The number of partition keys for this table. This gives the estimated number of partitions in

Name	Data Field	Unit	Data Type	Notes	Details
					the table.
Oauth2.tokens Partition Count	oauth2.tokens.estimatedpartition.count	N/A	Number		The number of partition keys for this table. This gives the estimated number of partitions in the table.
Oauth2.atokens Dropped Mutation	oauth2.atokens.dropped.mutation	N/A	String	This value should be 0.	The number of mutations (INSERTs, UPDATEs or DELETEs) started on this table but not completed. A large number of dropped mutations means that node is overloaded.
Oauth2.rtoken	oauth2.rtokens.dropped.muta	N/A	String	This value	The

Name	Data Field	Unit	Data Type	Notes	Details
ens Dropped Mutation	tion			should be 0.	number of mutations (INSERTs, UPDATEs or DELETEs) started on this table but not completed. A large number of dropped mutations means that node is overloaded.

NoSql Node Info

```
{
  "time": {epoch_time},
  "message": {
    "tag": "tml-nosql.{HOSTNAME}.metrics.cassandra.nodeinfo",
    "ingestion_time": "{Ingestion_time_to_log_service}",
    "node.datacentre": "dc1",
    "node.key_cache_rate": 0.925,
    "node.exception_count": 0,
    "node.load": "218.98KiB",
    "node.percentage_repaired": "100.0%",
    "node.heap_used": 135.25,
    "node.heap_total": 502.0
  }
}
```

Name	Data Field	Unit	Data Type	Details
Node Key Cache Rate	node.key_cache_rate	%	Number	Fraction of read requests for which the key's location on disk was found in the cache. The key cache hit rate provides visibility into the effectiveness of key cache. If the key cache hit rate is consistently high (above 0.85, or 85 percent), then the vast majority of read requests are being expedited through caching. If the key cache is not consistently serving up row locations for your read requests, consider increasing the size of the cache, which can be a low-overhead tactic for improving read latency.
Node Info Exception Count	node.exception_count	N/A	Number	Number exception thrown by Cassandra while reading/writing request. This number should be low or near to 0. If the exceptions count is increasing then type of exception needs to be find out. For instance, Cassandra's timeout exception reflects the incomplete (but not failed) handling of a

Name	Data Field	Unit	Data Type	Details
				request. Timeouts occur when the coordinator node sends a request to a replica and does not receive a response within the configurable timeout window. Timeouts are not necessarily fatal-the coordinator will store the update and attempt to apply it later-but they can indicate network issues or even disks nearing capacity. A more worrisome exception type is the unavailable exception, which indicates that Cassandra was unable to meet the consistency requirements for a given request, usually because one or more nodes were reported as down when the request arrived. For instance, in a cluster with replication factor of three and a consistency level of ALL, a read or write request will need to reach all three replica nodes in the cluster to perform a successful read or write.
Node Load	node.load	KIB/MiB	String	Disk space used on a

Name	Data Field	Unit	Data Type	Details
				node, in bytes. This number should always be less than total size allocated to Cassandra node.
Node Percent Repair	node.percentage_repaired	%	String	Since Cassandra fully embraces eventually consistency repair is actually a important mechanism for making sure copies of data are shipped around the cluster to meet your specified replication factor. If it is less then 50%, then it means that 50% of data is not replicated properly and hence we may end up losing the data.
Node Heap Usage	node.heap_used	N/A	Number	Total heap memory consumed by Cassandra Node.
Node Total Heap Allocation	node.heap_total	N/A	Number	Total heap memory allocated to Cassandra Node.

Diagnostic Recipe and Alerts for NoSQL

Metrics	Computation	Notes
Memory Usage	$(\text{node.heap\_used} * 100) / \text{node.heap\_total}$	Ideal is 50%. This indicates available buffer for high volume writes

Cassandra TP stats

```
{
  "time": {epoch_time},
  "message": {
    "tag": "tml-nosql.{HOSTNAME}.metrics.cassandra.tpstats",
    "ingestion_time": "{Ingestion_time_to_log_service}",
    "ReadStage.active": 0,
    "ReadStage.pending": 0,
    "ReadStage.completed": 7775,
    "ReadStage.blocked": 0,
    "ReadStage.all_time_blocked": 0,
    "MiscStage.active": 0,
    "MiscStage.pending": 0,
    "MiscStage.completed": 0,
    "MiscStage.blocked": 0,
    "MiscStage.all_time_blocked": 0,
    "CompactionExecutor.active": 0,
    "CompactionExecutor.pending": 0,
    "CompactionExecutor.completed": 8418,
    "CompactionExecutor.blocked": 0,
    "CompactionExecutor.all_time_blocked": 0,
    "MutationStage.active": 0,
    "MutationStage.pending": 0,
    "MutationStage.completed": 1864,
    "MutationStage.blocked": 0,
    "MutationStage.all_time_blocked": 0,
    "MemtableReclaimMemory.active": 0,
    "MemtableReclaimMemory.pending": 0,
    "MemtableReclaimMemory.completed": 372,
    "MemtableReclaimMemory.blocked": 0,
    "MemtableReclaimMemory.all_time_blocked": 0,
    "PendingRangeCalculator.active": 0,
    "PendingRangeCalculator.pending": 0,
    "PendingRangeCalculator.completed": 1,
    "PendingRangeCalculator.blocked": 0,
    "PendingRangeCalculator.all_time_blocked": 0,
    "GossipStage.active": 0,
    "GossipStage.pending": 0,
    "GossipStage.completed": 0,
    "GossipStage.blocked": 0,
    "GossipStage.all_time_blocked": 0,
    "SecondaryIndexManagement.active": 0,
    "SecondaryIndexManagement.pending": 0,
    "SecondaryIndexManagement.completed": 0,
  }
}
```

```

"SecondaryIndexManagement.blocked": 0,
"SecondaryIndexManagement.all_time_blocked": 0,
"HintsDispatcher.active": 0,
"HintsDispatcher.pending": 0,
"HintsDispatcher.completed": 0,
"HintsDispatcher.blocked": 0,
"HintsDispatcher.all_time_blocked": 0,
"RequestResponseStage.active": 0,
"RequestResponseStage.pending": 0,
"RequestResponseStage.completed": 20,
"RequestResponseStage.blocked": 0,
"RequestResponseStage.all_time_blocked": 0,
"Native-Transport-Requests.active": 0,
"Native-Transport-Requests.pending": 0,
"Native-Transport-Requests.completed": 45130,
"Native-Transport-Requests.blocked": 0,
"Native-Transport-Requests.all_time_blocked": 0,
"ReadRepairStage.active": 0,
"ReadRepairStage.pending": 0,
"ReadRepairStage.completed": 0,
"ReadRepairStage.blocked": 0,
"ReadRepairStage.all_time_blocked": 0,
"CounterMutationStage.active": 0,
"CounterMutationStage.pending": 0,
"CounterMutationStage.completed": 0,
"CounterMutationStage.blocked": 0,
"CounterMutationStage.all_time_blocked": 0,
"MigrationStage.active": 0,
"MigrationStage.pending": 0,
"MigrationStage.completed": 63,
"MigrationStage.blocked": 0,
"MigrationStage.all_time_blocked": 0,
"MemtablePostFlush.active": 0,
"MemtablePostFlush.pending": 0,
"MemtablePostFlush.completed": 723,
"MemtablePostFlush.blocked": 0,
"MemtablePostFlush.all_time_blocked": 0,
"PerDiskMemtableFlushWriter_0.active": 0,
"PerDiskMemtableFlushWriter_0.pending": 0,
"PerDiskMemtableFlushWriter_0.completed": 248,
"PerDiskMemtableFlushWriter_0.blocked": 0,
"PerDiskMemtableFlushWriter_0.all_time_blocked": 0,
"ValidationExecutor.active": 0,
"ValidationExecutor.pending": 0,
"ValidationExecutor.completed": 0,

```

```

"ValidationExecutor.blocked": 0,
"ValidationExecutor.all_time_blocked": 0,
"Sampler.active": 0,
"Sampler.pending": 0,
"Sampler.completed": 0,
"Sampler.blocked": 0,
"Sampler.all_time_blocked": 0,
"MemtableFlushWriter.active": 0,
"MemtableFlushWriter.pending": 0,
"MemtableFlushWriter.completed": 285,
"MemtableFlushWriter.blocked": 0,
"MemtableFlushWriter.all_time_blocked": 0,
"InternalResponseStage.active": 0,
"InternalResponseStage.pending": 0,
"InternalResponseStage.completed": 0,
"InternalResponseStage.blocked": 0,
"InternalResponseStage.all_time_blocked": 0,
"ViewMutationStage.active": 0,
"ViewMutationStage.pending": 0,
"ViewMutationStage.completed": 0,
"ViewMutationStage.blocked": 0,
"ViewMutationStage.all_time_blocked": 0,
"AntiEntropyStage.active": 0,
"AntiEntropyStage.pending": 0,
"AntiEntropyStage.completed": 0,
"AntiEntropyStage.blocked": 0,
"AntiEntropyStage.all_time_blocked": 0,
"CacheCleanupExecutor.active": 0,
"CacheCleanupExecutor.pending": 0,
"CacheCleanupExecutor.completed": 0,
"CacheCleanupExecutor.blocked": 0,
"CacheCleanupExecutor.all_time_blocked": 0
}
}

```

Onprem Loader Monitoring

The Onprem Loader syncs with Boomi Cloud API Management and loads data into the on-premises database in Local Edition. The sync process syncs the following:

- Developer classes
- Packages
- Services

- HttpClientSecurity

The success message shows number of records for each type.

Onprem Loader Success message example

```
{
  "tag": "tml-sql.mysql-set-0-0.sqldata",
  "ingestion_time": "2019-04-17T11:45:02+00:00",
  "process": "javaproxy",
  "function": "proxy_debug",
  "TML_CLUSTER_NAME": "Local Edition Reference Cluster",
  "TML_ZONE_NAME": "us-west1-a",
  "POD_IP": "10.8.1.9",
  "message": "[INFO] [pool-2-thread-1] c.m.p.data.management.OnpremLoader Loaded 10
records of HttpClientSecurity successfully. "
}
```

Onprem Loader Error message example

```
{
  "tag": "tml-sql.mysql-set-0-0.sqldata",
  "ingestion_time": "2019-04-16T16:30:01+00:00",
  "process": "javaproxy",
  "function": "proxy_error",
  "TML_CLUSTER_NAME": "Local Edition Reference Cluster Pune",
  "TML_ZONE_NAME": "us-west1-a",
  "POD_IP": "x.x.1.9",
  "message": "[ERROR] [pool-2-thread-1] c.m.p.data.management.OnpremLoader MOM
responded with 504 GATEWAY_TIMEOUT
com.mashery.proxy.data.exception.OperationException: MOM responded with 504 GATEWAY_
TIMEOUT \
    at com.mashery.proxy.data.management.OnpremLoader.doLoadKeysFromMom
(OnpremLoader.java: 1190) \
    at com.mashery.proxy.data.management.OnpremLoader.loadKeys(OnpremLoader.java:461)
\
    at
com.mashery.proxy.data.management.OnpremLoaderManager$LoaderTask.runInterruptibly
(OnpremLoaderManager.java:418) \
    at com.mashery.proxy.data.management.OnpremLoaderManager$LoaderTask.run
(OnpremLoaderManager.java:377) \
    at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149) \
    at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624) \
    at java.lang.Thread.run(Thread.java:748)"
}
```

Onprem Loader Error message example 2

```
{
  "tag": "tml-sql.mysql-set-0-0.sqldata",
  "ingestion_time": "2019-04-12T08:05:21+00:00",
  "process": "javaproxy",
  "function": "proxy_error",
  "TML_CLUSTER_NAME": "Local Edition Reference Cluster Pune",
  "TML_ZONE_NAME": "us-west1-a",
  "POD_IP": "x.x.1.9",
  "message": "[ERROR] [pool-2-thread-1] c.m.p.data.management.OnpremLoader
    javax.net.ssl.SSLException: SSL peer shut down incorrectly javax.net.ssl.SSLException:
SSL peer shut down incorrectly \
    at sun.security.ssl.InputRecord.readV3Record(InputRecord.java:596) \
    at sun.security.ssl.InputRecord.read(InputRecord.java:532) \
    at sun.security.ssl.SSLSocketImpl.readRecord(SSLSocketImpl.java:975) \
    at sun.security.ssl.SSLSocketImpl.readDataRecord(SSLSocketImpl.java:933) \
    at sun.security.ssl.AppInputStream.read(AppInputStream.java:105)"
}
```

Diagnostic Recipe and Alerts for Onprem Loader

Metrics	Recipe	Notes
message	log["message"].contains("[INFO]") and log ["message"].contains("Loaded 10 records of HttpsClientSecurity successfully ")	Similar messages with mapis, devclasses, httpclientsecurity, packages

Memcached Loader Monitoring

The Memcached Loader is responsible for updating the following in the different Memcached pools:

- Mapis
- Developer classes
- Services
- Package keys
- HttpClientSecurity

The loader updates delta changes within last ten minutes.

Memcached Loader - Success message example

```
{
  "tag": "tml-cache.cache-set-0-1",
  "ingestion_time": "2019-04-12T08:10:26+00:00",
  "process": "javaproxy",
  "function": "proxy_debug",
  "TML_CLUSTER_NAME": "Local Edition Reference Cluster Pune",
  "TML_ZONE_NAME": "us-west1-a",
  "POD_IP": "x.x.0.13",a
  "message": "[INFO ] [pool-5-thread-1] c.m.p.memcacheloader.MemcacheLoader Loading services.
"
}
{
  "tag": "tml-cache.cache-set-0-1",
  "ingestion_time": "2019-04-12T08:10:27+00:00",
  "process": "javaproxy",
  "function": "proxy_debug",
  "TML_CLUSTER_NAME": "Local Edition Reference Cluster Pune",
  "TML_ZONE_NAME": "us-west1-a",
  "POD_IP": "x.x.0.13",
  "message": "[INFO ] [pool-5-thread-1] c.m.p.i.m.MemcachedClientPool Active Memcached Clients:
1 "
}
{
  "tag": "tml-cache.cache-set-0-1",
  "ingestion_time": "2019-04-12T08:11:23+00:00",
  "process": "javaproxy",
  "function": "proxy_debug",
  "TML_CLUSTER_NAME": "Local Edition Reference Cluster Pune",
  "TML_ZONE_NAME": "us-west1-a",
  "POD_IP": "x.x.0.13",
  "message": "[INFO ] [pool-5-thread-1] c.m.p.memcacheloader.MemcacheLoader Successfully
loaded Services. ServicesCount: 683 "
}
```

Memcached Loader error example

```
{
  "tag": "tml-cache.cache-set-0-2",
  "ingestion_time": "2019-04-17T11:55:01+00:00",
  "process": "javaproxy",
  "function": "proxy_error",
  "TML_CLUSTER_NAME": "Local Edition Reference Cluster",
  "TML_ZONE_NAME": "us-west1-a",
  "POD_IP": "x.x.2.8",
```

"message":"[ERROR] [pool-5-thread-2] c.m.p.memcacheloader.MemcacheLoader Unable to load mapis.

```
java.net.ConnectException: Connection refused (Connection refused) \
  at java.net.PlainSocketImpl.socketConnect(Native Method) \
  at java.net.AbstractPlainSocketImpl.doConnect(AbstractPlainSocketImpl.java:350) \
  at java.net.AbstractPlainSocketImpl.connectToAddress(AbstractPlainSocketImpl.java:206) \
  at java.net.AbstractPlainSocketImpl.connect(AbstractPlainSocketImpl.java:188) \
  at java.net.SocksSocketImpl.connect(SocksSocketImpl.java:392) \
  at java.net.Socket.connect(Socket.java:589) \
  at java.net.Socket.connect(Socket.java:538) \
  at java.net.Socket.<init>(Socket.java:434) \
  at java.net.Socket.<init>(Socket.java:244) \
  at com.mysql.jdbc.StandardSocketFactory.connect(StandardSocketFactory.java:258) \
  at com.mysql.jdbc.MySQLIO.<init>(MySQLIO.java:305)
Wrapped by: com.mysql.jdbc.exceptions.jdbc4.CommunicationsException: Communications link
failure The last packet sent successfully to the server was 0 milliseconds ago. The driver has not
received any packets f
rom the server. \
  at sun.reflect.NativeConstructorAccessorImpl.newInstance0(Native Method) \
  at sun.reflect.NativeConstructorAccessorImpl.newInstance
(NativeConstructorAccessorImpl.java:62) \
  at sun.reflect.DelegatingConstructorAccessorImpl.newInstance
(DelegatingConstructorAccessorImpl.java:45) \
  at java.lang.reflect.Constructor.newInstance(Constructor.java:423) \
  at com.mysql.jdbc.Util.handleNewInstance(Util.java:408) \
  at com.mysql.jdbc.SQLError.createCommunicationsException(SQLError.java:1137) \
  at com.mysql.jdbc.MySQLIO.<init>(MySQLIO.java:355) \
  at com.mysql.jdbc.ConnectionImpl.coreConnect(ConnectionImpl.java:2490) \
  at com.mysql.jdbc.ConnectionImpl.connectOneTryOnly(ConnectionImpl.java:2527) \
  at com.mysql.jdbc.ConnectionImpl.createNewIO(ConnectionImpl.java:2309) \
  at com.mysql.jdbc.ConnectionImpl.<init>(ConnectionImpl.java:834) \
  at com.mysql.jdbc.JDBC4Connection.<init>(JDBC4Connection.java:46) \
  at sun.reflect.GeneratedConstructorAccessor18.newInstance(Unknown Source) \
  at sun.reflect.DelegatingConstructorAccessorImpl.newInstance
(DelegatingConstructorAccessorImpl.java:45) \
  at java.lang.reflect.Constructor.newInstance(Constructor.java:423) \
  at com.mysql.jdbc.Util.handleNewInstance(Util.java:408) \
  at com.mysql.jdbc.ConnectionImpl.getInstance(ConnectionImpl.java:419) \
  at com.mysql.jdbc.NonRegisteringDriver.connect(NonRegisteringDriver.java:344) \
  at com.mysql.jdbc.jdbc2.optional.MysqlDataSource.getConnection(MysqlDataSource.java:441) \
  at com.mysql.jdbc.jdbc2.optional.MysqlDataSource.getConnection(MysqlDataSource.java:139) \
  at com.mysql.jdbc.jdbc2.optional.MysqlDataSource.getConnection(MysqlDataSource.java:109) \
  at com.mysql.jdbc.jdbc2.optional.MysqlConnectionPoolDataSource.getPooledConnection
(MysqlConnectionPoolDataSource.java:60) \
  at
```



```

com.mashery.proxy.internal.datasource.MonitorableMysqlConnectionPoolDataSource.getPooled
Connection(MonitorableMysqlConnectionPoolDataSource.java:41) \
    at org.apache.commons.dbcp.datasources.KeyedCPDSCConnectionFactory.makeObject
(KeyedCPDSCConnectionFactory.java:127) \
    at org.apache.commons.pool.impl.GenericKeyedObjectPool.borrowObject
(GenericKeyedObjectPool.java:1212) \
    at
org.apache.commons.dbcp.datasources.SharedPoolDataSource.getPooledConnectionAndInfo
(SharedPoolDataSource.java:181)
Wrapped by: org.apache.commons.dbcp.SQLNestedException: Could not retrieve connection info
from pool \
    at
org.apache.commons.dbcp.datasources.SharedPoolDataSource.getPooledConnectionAndInfo
(SharedPoolDataSource.java:184) \
    at org.apache.commons.dbcp.datasources.InstanceKeyDataSource.getConnection
(InstanceKeyDataSource.java:701) \
    at org.apache.commons.dbcp.datasources.InstanceKeyDataSource.getConnection
(InstanceKeyDataSource.java:676) \
    at com.mashery.proxy.datasource.mysql.MySQLSharedPoolDataSource.getConnection
(MySQLSharedPoolDataSource.java:129) \
    at com.mashery.proxy.memcacheloader.MemcacheLoader.doLoadMapis
(MemcacheLoader.java:1227) \
    at com.mashery.proxy.memcacheloader.MemcacheLoader.loadMapis
(MemcacheLoader.java:1203) \
    at
com.mashery.proxy.internal.memcacheloader.MemcacheLoaderManager$LoaderTask.runInterru
ptibly(MemcacheLoaderManager.java:456) \
    at com.mashery.proxy.internal.memcacheloader.MemcacheLoaderManager$LoaderTask.run
(MemcacheLoaderManager.java:381) \
    at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149) \
    at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624) \
    at java.lang.Thread.run(Thread.java:748)"}

```

Diagnostic Recipe and Alerts for Memcached Loader

Metric	Recipe	Notes
Message	log["message"].contains("[INFO]") and log["message"].contains("Successfully loaded services")	Similar messages with mapis, devclasses, httpclientsecurity, packages.

Functional Monitoring

Functional monitoring includes watching:

- Response codes other than 200, such 403, 5xx.
- Response times

Functional Monitoring is about detecting sudden or sustained spikes in either the response codes and or response times.

There could be a myriad of reasons why the health of Local Edition can deteriorate.

What to Monitor	Thresholds	Runbook	Comments
API calls are returning 596	Low watermark: repeated 596 in a low interval say 1 min High watermark: Repeated 596 for many APIs in low interval say 1min OnPrem missed 2 or more cycles in a row. OnPrem loader failure on each of the two invocations.	Check Memcache Stats, OnPremLoader and Memcache Loader Status	Possible reasons: 1. Memcache is evicting because of slab fragmentation. 2. Onprem loader has not successfully synced with MOM. 3. Onprem loader could not update SQL DB. 4. Memcache loader has not picked up updated data from SQL.
High latency in API traffic	Low watermark: SLA breached within 30 seconds for API calls High watermark: Regular & Repeated Latency spikes	Check NoSQL Stats, Memcache Stats, TM Stats	Possible reasons: 1. OAuth tokens, keys validation is slow. 2. Memcache

What to Monitor	Thresholds	Runbook	Comments
	Continuous prolonged high latency for a long duration of say 5 mins		retrieval is slow. 3. Target System is slow to respond. 4. The request is queued due to unavailability of threads on Proxy.
APU calls are returning 403	Low watermark: repeated 403 with tokens inside a 30-second interval High watermark: prolonged 403 or regular occurrence of the low watermark		Possible reasons 403 errors are typically seen with OAuth token validation and this problem will be typically seen in a multi-zone setup due to dependency on Cassandra replicating across zones.
Continuous Monitoring- Running a pre-configured Protected / Unprotected Traffic Call for each TM every x min.			Protected Call on each TM - The call should include creating a new token, making a traffic call, deleting a token.

Recipes for ELK

Memcached Recipes

Metric	Recipe	Notes
Hit rate	Script Field: HIT Rate <pre>doc['get_hits'].value/ ((doc['get_hits'].value+ doc['get_misses'].value)>0? (doc['get_hits'].value +doc['get_misses'].value):1)</pre>	Define an alter for a condition where Hit rate > 0 Example: <pre>tag:tmgc-cache.metrics.stat.memcached.11214 and memcached_hit_rate > 0</pre>

ELK Tuning

ELK message getting dropped from Log Service due to limitations on fields per index

If an index has more than 1000 separate fields, then Elasticsearch doesn't accept the new messages. This is due to default setting on the index at Elasticsearch side. Increase the number of fields for `metrics_index` in order to get all the logs in ELK stack.

```
curl -XPUT '{ElasticSearch_hostname}:9200/{metrics_index}/_settings' -H 'Content-Type:
application/json' -d'
{
  "index": {
    "mapping": {
      "total_fields": {
        "limit": "10000"
      }
    }
  }
}
```

Connectors and Adapters

Connectors are Boomi Cloud API Management's integration features that have been developed and provided ready to use for Cloud customers. Connectors have been carefully envisioned to address common use-cases, such as: content injection, content filtering, content transformation, call authentication using third-party IDP, IP-based call filtering, domain-based routing, geo-location based routing and HTTP header manipulation.

Boomi Cloud API Management-built Connector vs Adapter

The following table explains the differences between Boomi Cloud API Management-built connectors and Boomi Cloud API Management-built adapters.

Connector	Adapter
Boomi-built features for Boomi Cloud API Management	Boomi-built for both Boomi Cloud API Management and Local Edition
Strategic integration feature generally available out of the box for any Boomi Cloud API Management customer.	Custom in nature and tailored to fit a particular customer specific need.
Available without additional cost to Boomi Cloud API Management customers.	Customer may incur fee for engineering and development work done by Boomi.
	Note: Local Edition customers can use the SDK to build their own adapters.
Can be ported to a Local Edition customer upon request in Mashery Local 4.X.	Needs to be engineered and developed for Local Edition customer.

FAQs

Does Boomi ship Boomi Cloud API Management Connectors out of the box?

Yes. Boomi Cloud API Management Connectors are available out of the box for Boomi Cloud API Management customers.

Can Boomi Cloud API Management Connectors be made available or ported to Local Edition customers?

Yes. Boomi Cloud API Management built Connectors can be ported to Local Edition customers upon request.

How can I get a Boomi Cloud API Management built certified Connector?

In order to get Boomi Cloud API Management Connectors ported, a Local Edition customer needs to make request to the Boomi Support team.

Boomi will release it to the Local Edition customer specific area. Once Boomi Cloud API Management Connectors are promoted to customer specific area, they will be available to deploy via Local Edition user interface. For more information see "How to Get the Boomi Cloud API Management-Built Connectors and/or Adapters" topic.

Why aren't Boomi Cloud API Management Connectors available out of the box to prospects/leads?

- Boomi Cloud API Management built Connectors are packaged and ported specifically to the Local Edition customer area.
- Boomi prospects/leads do not get assigned a Boomi Cloud API Management production area unless they are formally on-boarded in Boomi Cloud API Management.

i Note: In certain cases, prospects/leads may get a trial area, but one that is temporarily assigned for evaluation period.

What are some best practices for a Local Edition customer who needs to migrate to newer version of Local Edition and consumes Boomi Cloud API Management built Adapters and/or Connectors?

Here are best practices and guidance for smoother migration to newer version of Local Edition 4.4.X.

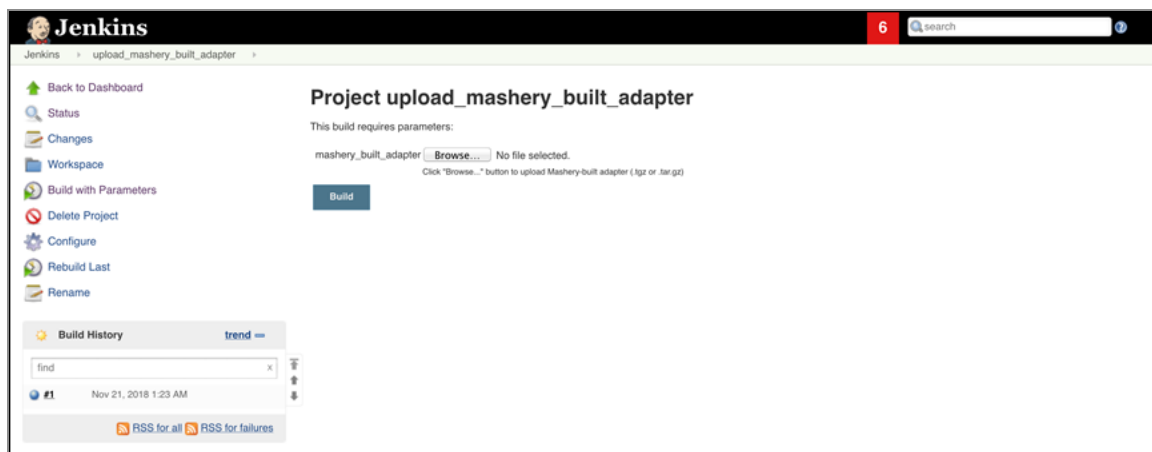
i Note: (Applies to customers who consume Boomi Cloud API Management built Connectors or Adapters)

Normal Flow

1. It is always good practice and Boomi recommends to release Boomi Cloud API Management built Connectors or Adapters certified for a particular Local Edition 4.4.X release in a non-production area/environment first.
 - a. Migration of Local Edition product and Boomi Cloud API Management built Connectors or Adapters that is needed for a particular Local Edition 4.4.X version must happen in lock step.
2. Customer should deploy Boomi Cloud API Management built Connectors or Adapters build once they are available in their non-production environment for a particular Local Edition 4.4.X version.
3. Customer must perform UAT and confirms/sign off that Boomi Cloud API Management built Connectors or Adapters are functioning as expected for Local Edition version.
4. Customer should update support case once they sign off UAT. Boomi will release Connectors or Adapters to customer Local Edition production area.

Boomi-provided Connectors and Adapters

Jenkins job "upload-mashery-built-adapter" is for uploading Boomi Cloud API Management-built connectors and adapters, it can be accessed using the Jenkins web console:



For instructions on how to upload a Boomi Cloud API Management-built connector or adapter, refer to [Prebuilding](#).

Boomi Cloud API Management Connectors

Connectors are Boomi Cloud API Management's integration features that have been developed and provided ready to use for Cloud customers. Connectors not only provide connectivity to the external systems but also allow you to change the behavior of the API at runtime through configuration without writing any code. A Connector's policies can be applied for pre-processing of user client request as well as for post processing on backend response to change API behavior. Connectors have been carefully envisioned to address common and evolving business use-cases for customers. Connectors power compelling features such as HTTP header manipulation, authentication & security (also using third-party IDP), filtering & routing, caching & validation, as well as content transformation.

Connectors can provide the following benefits:

- It complements Boomi Cloud API Management product core functionality and fills gap for commonly-sought functionality that are not yet available out-of-box.
- It can be used across multiple Boomi Cloud API Management customers who need similar functionality.
- Further enhances Boomi Cloud API Management's competitive edge and relevancy to the customers/markets' continuously-evolving business needs.

Contact the Boomi Customer Success team if you are interested to know more whether Boomi Cloud API Management Connectors meets your business needs and how to get it.

For more information, see [Boomi Cloud API Management Connectors](#).

Verifying the Boomi Cloud API Management Connectors and Adapters

To verify if the Boomi Cloud API Management Connectors and Adapters were loaded and deployed successfully perform the following steps:

Procedure

1. Check the status of all the containers as follows:


```
njaltare@appss-MacBook-Pro-5 manifest-onprem-swarm % docker ps -a
CONTAINER ID   IMAGE                                COMMAND                  CREATED        STATUS
PORTS         NAMES
0d847c71a32c   tml-tm:v5.3.1.GA.3                "/usr/local/bin/tm-e..." 4 hours ago    Up 4 hours
(healthy)     22/tcp, 80/tcp, 443/tcp, 8000/tcp, 8083-8084/tcp  tmstack_
tml.1.nexbrmw2np59yky95mm8k4tm6
dbfd3ae614fc   tml-cache:v5.3.1.GA.3             "/usr/local/bin/cach..." 4 hours ago    Up 4 hours
11211-11216/tcp  cachestack_cache.1.uprbqvq472ckrrl4l6yy49if4
0bf0e830f261   tml-sql:v5.3.1.GA.3               "/usr/local/bin/sql-..." 4 hours ago    Up 4 hours
3306/tcp      sqlstack_sql.1.sxhask7woay9d9xtufv911wi
352d9e5cb78d   tml-log:v5.3.1.GA.3               "/usr/local/bin/log-..." 4 hours ago    Up 4 hours
24023/tcp, 24220/tcp, 24224/tcp  logstack_log.1.f75ej5wha5t7htjt2y27wg8ie
9b5de6dd0847   tml-cm:v5.3.1.GA.3                "/usr/local/bin/cm-e..." 4 hours ago    Up 4 hours
(healthy)     cmstack_tmlcm.1.ymgtad7ty8to6606s29cjspxvj
87496d0e2a4c   tml-nosql:v5.3.1.GA.3             "/usr/local/bin/nosq..." 4 hours ago    Up 4 hours
(healthy)     7000-7001/tcp, 7199/tcp, 9042/tcp, 9160/tcp  nosqlstack_nosql_
seed.1.ebce8f5eptjlib651wyg15qhr
```

All the containers namely, tml-tm, tml-cache, tml-sql, tml-log, tml-cm and tml-nosql must be up and running.

2. Login to the Traffic Manager container tml-tm as follows:

```
njaltare@appss-MacBook-Pro-5 manifest-onprem-swarm % docker exec -it 0d847c71a32c
bash
[root@0d847c71a32c builder]#
```

3. On successfully logging into the Traffic Manager container, you can access the OSGi console of the Traffic Manager using the following command:

```
[root@0d847c71a32c builder]# telnet localhost 8081
```

4. Verify if connection to the OSGi console is successful as follows:

```
[root@0d847c71a32c builder]# telnet localhost 8081
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
OSGi>
```

5. List the OSGi bundles by executing the ss command.

An example of OSGi ss command is shown below.

```

OSGi> ss
"Framework is launched."

id State   Bundle
0  ACTIVE   org.eclipse.OSGi_3.15.0.v20190830-1434
1  ACTIVE   org.eclipse.equinox.simpleconfigurator_1.3.300.v20190716-0825
2  RESOLVED  ch.qos.logback.access_1.1.11
3  RESOLVED  ch.qos.logback.classic_1.1.11
          Fragments=42
4  RESOLVED  ch.qos.logback.core_1.1.11
5  RESOLVED  ch.qos.logback.slf4j_1.1.2.v20160301-0943
          Master=147
6  RESOLVED  com.amazonaws_1.1.9
7  RESOLVED  com.codahale.metrics.core_3.1.2
8  RESOLVED  com.datastax.driver.core_3.7.2
9  RESOLVED  com.google.guava_27.1.0.v20190517-1946
10 ACTIVE   com.mashery.cassandra.client_1.0.0.v20200122-1721
11 STARTING com.mashery.geoip_1.0.0.v20200122-1721
12 RESOLVED com.mashery.http_1.2.0.v20191113-0329
13 ACTIVE   com.mashery.internal.http_1.0.0.v20200130-0430
14 RESOLVED com.mashery.internal.util_1.0.1.v20200329-0041
15 STARTING com.mashery.mail.aws_1.0.0.v20200122-1721
16 STARTING com.mashery.mail.log_1.0.0.v20200313-2203
17 RESOLVED com.mashery.oauth2_1.2.0.v20200311-2141
18 ACTIVE   com.mashery.perf_1.0.0.r51209
19 ACTIVE   com.mashery.perf.metric_1.0.0.v20180828-0655
20 STARTING com.mashery.perf.statsd_1.0.0.r51209
21 RESOLVED com.mashery.proxy.config_1.0.0.v20200323-2237
22 ACTIVE   com.mashery.proxy.configurator_1.0.0.v20200130-0430
23 ACTIVE   com.mashery.proxy.connector.oauth2jwtconnector_1.1.0.v20200330-1631
24 STARTING com.mashery.proxy.connectors.util_1.1.0.v20200330-1631
25 ACTIVE   com.mashery.proxy.core_1.0.0.v20200222-0620
26 ACTIVE   com.mashery.proxy.core.crypto_1.0.0.v20200218-2209
27 ACTIVE   com.mashery.proxy.core.oauth1_1.0.0.v20200122-1721
28 ACTIVE   com.mashery.proxy.core.oauth2_1.2.0.v20200312-0442
29 RESOLVED com.mashery.proxy.crypto_1.0.0.v20200130-0430
30 STARTING com.mashery.proxy.customer.common.xmltojson_1.2.0.v20200402-0713
31 ACTIVE   com.mashery.proxy.customer.generic_1.2.0.v20200529-0649
32 ACTIVE   com.mashery.proxy.customers.util_1.2.0.v20200526-1123
33 STARTING com.mashery.proxy.dataservice_1.1.0.v20200313-2203
34 ACTIVE   com.mashery.proxy.datasource_1.1.0.v20200406-0749
35 ACTIVE   com.mashery.proxy.datasource.compatibility_1.0.0.v20200130-0430
36 ACTIVE   com.mashery.proxy.dataservice_1.0.0.v20180828-0655

```

```

37 RESOLVED com.mashery.proxy.email_2.0.0.v20200319-0236
38 ACTIVE com.mashery.proxy.email.provider_1.0.0.v20200122-1721
39 ACTIVE com.mashery.proxy.env_1.0.0.v20200406-0749
40 RESOLVED com.mashery.proxy.launcher_1.1.0.v20200309-2140
41 ACTIVE com.mashery.proxy.logback_1.0.0.v20200122-1721
42 RESOLVED com.mashery.proxy.logback.config_1.0.0.v20200122-1721
    Master=3
43 RESOLVED com.mashery.proxy.logback.syslog_1.0.0.v20200122-1721
44 ACTIVE com.mashery.proxy.management_1.0.0.v20200320-2008
45 ACTIVE com.mashery.proxy.memcached_1.0.1.v20200317-2011
46 ACTIVE com.mashery.proxy.memcacheloader_1.1.0.v20200305-1624
47 RESOLVED com.mashery.proxy.oauth2_1.1.0.v20180731-2352
48 ACTIVE com.mashery.proxy.oauth2.onprem_1.0.0.v20200327-1828
49 ACTIVE com.mashery.proxy.onprem.crypto_1.0.0.v20200130-0430
50 ACTIVE com.mashery.proxy.onprem.jmx_1.0.0.v20200218-2209
51 RESOLVED com.mashery.proxy.onprem.mom.msg_1.0.0.v20200320-0726
52 ACTIVE com.mashery.proxy.onprem.tokenpurge_1.0.0.v20200122-1721
53 ACTIVE com.mashery.proxy.onpremloader_1.1.0.v20200218-2209
54 ACTIVE com.mashery.proxy.OSGi.console.cmd_1.0.0.v20200218-2209
55 RESOLVED com.mashery.proxy.OSGi.hooks_1.1.0.v20200225-0641
56 ACTIVE com.mashery.proxy.responder_1.2.0.v20200327-1828
57 ACTIVE com.mashery.proxy.server_1.1.0.v20200327-1828
58 ACTIVE com.mashery.proxy.servlet_1.1.0.v20200410-2009
59 ACTIVE com.mashery.proxy.signal_1.0.1.v20200222-0620
60 RESOLVED com.mashery.proxy.slf4j_1.0.0.v20140814-1844
    Master=147
61 STARTING com.mashery.proxy.slf4j.helper_1.0.0.v20200220-2055
62 ACTIVE com.mashery.proxy.verbose.logger_1.0.0.v20181112-1720
63 ACTIVE com.mashery.proxy.xml.parsers_1.0.0.v20200122-1721
64 ACTIVE com.mashery.service_1.3.0.v20200319-1949
65 ACTIVE com.mashery.service.oauth2_1.2.0.v20200221-2225
66 ACTIVE com.mashery.service.onprem_1.0.0.v20200218-2209
67 ACTIVE com.mashery.service.onprem.oauth_1.0.0.v20200218-2209
68 ACTIVE com.mashery.service.responder_1.1.0.v20200319-1949
69 RESOLVED com.mashery.signal_1.0.0.v20111004-1955
70 RESOLVED com.mashery.trafficmanager.sdk_1.2.0.v20200313-2203
71 RESOLVED com.mashery.util_1.1.0.v20200319-1810
72 RESOLVED com.springsource.org.objenesis_1.0.0
73 RESOLVED io.dropwizard.metrics.core_3.2.6
74 RESOLVED io.netty.buffer_4.1.46.Final
75 RESOLVED io.netty.codec_4.1.46.Final
76 RESOLVED io.netty.common_4.1.46.Final
77 RESOLVED io.netty.handler_4.1.46.Final

```

```

78 RESOLVED io.netty.resolver_4.1.46.Final
79 RESOLVED io.netty.transport_4.1.46.Final
80 RESOLVED jackson-core-asl_1.6.3
81 RESOLVED jackson-mapper-asl_1.6.3
82 RESOLVED javax.mail_1.4.4
83 RESOLVED javax.servlet_3.1.0.v201410161800
84 RESOLVED net.bytebuddy.byte-buddy_1.9.0.v20181107-1410
85 RESOLVED net.bytebuddy.byte-buddy-agent_1.9.0.v20181106-1534
86 RESOLVED net.jpountz.lz4_1.3.0
87 RESOLVED net.spy.memcached_2.12.3
88 RESOLVED org.apache.commons.cli_1.4.0
89 RESOLVED org.apache.commons.codec_1.10.0.v20180409-1845
90 RESOLVED org.apache.commons.collections_3.2.2
91 RESOLVED org.apache.commons.dbcp_1.4.0
92 RESOLVED org.apache.commons.httpclient_3.1.0
93 RESOLVED org.apache.commons.io_2.6.0.v20190123-2029
94 RESOLVED org.apache.commons.lang_2.6.0.v201404270220
95 RESOLVED org.apache.commons.lang3_3.9.0
96 RESOLVED org.apache.commons.logging_1.2.0.v20180409-1502
97 RESOLVED org.apache.commons.pool_1.6.0
98 ACTIVE   org.apache.felix.gogo.command_1.0.2.v20170914-1324
99 ACTIVE   org.apache.felix.gogo.runtime_1.1.0.v20180713-1646
100 ACTIVE  org.apache.felix.gogo.shell_1.1.0.v20180713-1646
101 ACTIVE  org.apache.felix.scr_2.1.14.v20190123-1619
102 RESOLVED org.apache.httpcomponents.httpclient_4.0.3
103 RESOLVED org.apache.httpcomponents.httpcore_4.1.0
104 RESOLVED org.apache.httpcomponents.httpmime_4.0.3
105 RESOLVED org.apache.james.mime4j_0.6.0
106 ACTIVE   org.eclipse.core.contenttype_3.7.400.v20190624-1144
107 ACTIVE   org.eclipse.core.jobs_3.10.500.v20190620-1426
108 ACTIVE   org.eclipse.core.runtime_3.16.0.v20190823-1314
109 ACTIVE   org.eclipse.equinox.app_1.4.300.v20190815-1535
110 ACTIVE   org.eclipse.equinox.cm_1.4.100.v20190714-1849
111 ACTIVE   org.eclipse.equinox.common_3.10.500.v20190815-1535
112 ACTIVE   org.eclipse.equinox.console_1.4.0.v20190819-1430
113 ACTIVE   org.eclipse.equinox.ds_1.6.0.v20190122-0806
114 ACTIVE   org.eclipse.equinox.event_1.5.200.v20190814-0953
115 ACTIVE   org.eclipse.equinox.http.jetty_3.7.200.v20190714-1849
116 RESOLVED org.eclipse.equinox.http.registry_1.1.700.v20190214-1948
117 ACTIVE   org.eclipse.equinox.http.servlet_1.6.200.v20190823-1423
118 INSTALLED org.eclipse.equinox.launcher.gtk.linux.x86_64_1.1.1100.v20190907-0426
119 ACTIVE   org.eclipse.equinox.preferences_3.7.500.v20190815-1535
120 ACTIVE   org.eclipse.equinox.registry_3.8.500.v20190714-1850

```

```

121 RESOLVED org.eclipse.equinox.supplement_1.9.0.v20190830-1434
122 RESOLVED org.eclipse.equinox.useradmin_1.1.700.v20181116-1551
123 STARTING org.eclipse.equinox.util_1.1.300.v20190714-1852
124 RESOLVED org.eclipse.equinox.wireadmin_1.0.800.v20180827-1235
125 RESOLVED org.eclipse.jetty.alpn.client_9.4.20.v20190813
126 RESOLVED org.eclipse.jetty.client_9.4.20.v20190813
127 RESOLVED org.eclipse.jetty.continuation_9.4.20.v20190813
128 RESOLVED org.eclipse.jetty.http_9.4.20.v20190813
129 RESOLVED org.eclipse.jetty.http2.client_9.4.20.v20190813
130 RESOLVED org.eclipse.jetty.http2.client.http_9.4.20.v20190813
131 RESOLVED org.eclipse.jetty.http2.common_9.4.20.v20190813
132 RESOLVED org.eclipse.jetty.http2.hpack_9.4.20.v20190813
133 RESOLVED org.eclipse.jetty.http2.server_9.4.20.v20190813
134 RESOLVED org.eclipse.jetty.io_9.4.20.v20190813
135 RESOLVED org.eclipse.jetty.OSGi-servlet-api_3.1.0.M3
136 RESOLVED org.eclipse.jetty.OSGi.httpservice_9.4.20.v20190813
137 RESOLVED org.eclipse.jetty.security_9.4.20.v20190813
138 RESOLVED org.eclipse.jetty.server_9.4.20.v20190813
139 RESOLVED org.eclipse.jetty.servlet_9.4.20.v20190813
140 RESOLVED org.eclipse.jetty.util_9.4.20.v20190813
141 RESOLVED org.eclipse.OSGi.services_3.8.0.v20190206-2147
142 RESOLVED org.eclipse.OSGi.util_3.5.300.v20190708-1141
143 RESOLVED org.mariadb.jdbc_2.5.4
144 RESOLVED org.mockito_2.23.0.v20190527-1420
145 RESOLVED org.objenesis_2.6.0.v20180420-1519
146 RESOLVED org.OSGi.service.jdbc_1.0.0.201505202023
147 RESOLVED org.slf4j.api_1.7.10.v20170428-1633
    Fragments=60, 5
148 RESOLVED org.slf4j.jcl_1.7.2.v20130115-1340
149 RESOLVED org.slf4j.log4j_1.7.2.v20130115-1340
150 RESOLVED OSGi.core_5.0.0.201203141902
151 RESOLVED OSGi.enterprise_4.2.0.v201108120515
152 RESOLVED slf4j.api_1.7.28

```

6. The Boomi Cloud API Management Connectors will be listed as a single bundle.

For example, the bundle for a custom adapter built with Boomi Cloud API Management Connectors and standalone OAuth2JWT Connector, is as follows:

id	State	Bundle
23	ACTIVE	com.mashery.proxy.connector.oauth2jwtconnector_1.1.0.v20200330-1631
31	ACTIVE	com.mashery.proxy.customer.generic_1.2.0.v20200529-0649
32	ACTIVE	com.mashery.proxy.customers.util_1.2.0.v20200526-1123

i Note:

- Boomi Cloud API Management Connectors bundle can have different version values based on your deployment. The value 1.2.0.v20200529-0649 can be different accordingly.
- The example shown above is of an OAuth2JWT Connector, com.mashery.proxy.connector.oauth2jwtconnector_1.1.0.v20200330-1631 deployed as standalone Connector bundle.

7. The Boomi Cloud API Management built Adapters, if any, will be listed as a single bundle.

For example, a bundle for a Boomi built adapter is as follows:

id	State	Bundle
31	ACTIVE	com.mashery.proxy.customer.acme_1.2.0.v20200529-0649



Note: Boomi Cloud API Management Adapter bundle can have different version values based on your deployment. The value 1.2.0.v20200529-0649 can be different accordingly.

8. Execute the exit command to exit from the OSI console.

Type exit and hit the enter button twice.

Sample output is as follows:

```
Login to TML-TM container bash.
njaltare@appss-MacBook-Pro-5 manifest-onprem-swarm % docker exec -it c038db019f96
bash
```

```
[root@c038db019f96 builder]# telnet localhost 8081Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
```

```
OSGi> quit
gogo: CommandNotFoundException: Command not found: quit
```

```
OSGi> exit
```

```
Really want to stop Equinox? (y/n; default=y)
Connection closed by foreign host.
```

```
[root@c038db019f96 builder]#
```

**Note:**

- quit is not supported command as an exit command from OSGi console.
- To log into the OSGi console again, follow the instructions in sequence above.
 - Make sure to exit completely from the TML-TM container.
 - Re-login to the container bash.
 - Login to OSGi again.

FAQs

Does Boomi ship Boomi Cloud API Management Connectors out of the box?

Yes. Boomi Cloud API Management Connectors are available out of the box for Boomi Cloud API Management customers.

Where do I get notification about new features powered by Boomi Cloud API Management Connectors?

As new features powered by Boomi Cloud API Management Connectors get released, these updates are communicated through the Boomi Cloud API Management Release Notes on the [Boomi Community site](#).

Can Boomi Cloud API Management Connectors be made available or ported to Local Edition customers?

Yes. Boomi Cloud API Management built Connectors can be ported to API Management - Local Edition upon customer request.

How can I get a Boomi Cloud API Management built certified Connector?

Boomi Cloud API Management Connectors can be ported by a Local Edition customer by making a request to the Boomi Support team.

Do Boomi Cloud API Management Connectors get ported to classic Local Edition as well as next-generation Local Edition?

Local Edition 4.x.x and earlier are considered classic, see [Local Edition 4.0.0](#).

Local Edition 5.x.x and later are considered next-generation, see [Local Edition 5.0.0](#).

Most Boomi Cloud API Management Connectors are ported to both classic Local Edition as well as next-generation Local Edition. Refer to the Local Edition Porting section for the specific Connector in the Boomi Cloud API Management Connector user guide for its porting and availability.

How do I deploy Boomi Cloud API Management built Connectors ported in Local Edition?

For classic Local Edition, see "How to Get the Boomi Cloud API Management-Built Adapter and/or Connector" in Local Edition Virtual Appliance Installation and Configuration Guide.

For next-generation Local Edition, see [Boomi-provided Adapters and Connectors](#).

How does Boomi deliver Boomi Cloud API Management Connectors for Local Edition 5_X_X?

For customers using only out of the box Boomi Cloud API Management connectors, Local Edition does not support porting of a single Boomi Cloud API Management connector yet. You can download the Boomi Cloud API Management connectors bundle artifact from [Boomi support](#) then preload the Connectors bundle artifact (.tgz) and then make the TM (Traffic Manager) integrated docker build in Local Edition 5_X_X.

Instructions on how to navigate to hot fixes as well as Boomi Cloud API Management Connectors for Local Edition 5_X_X artifact are available at [Boomi hotfix support](#).

For Local Edition 5.3.0 or earlier versions the connectors are located at MasheryLocal > 5.2 > Connectors_20190923R4-tar.gz.

For Local Edition 5.3.1 or newer versions the latest build are available at MasheryLocal > TIBCO_Mashery_Built_Connectors > 20200529 > Mashery_Connectors_20200529R4.tgz from [Boomi support](#).

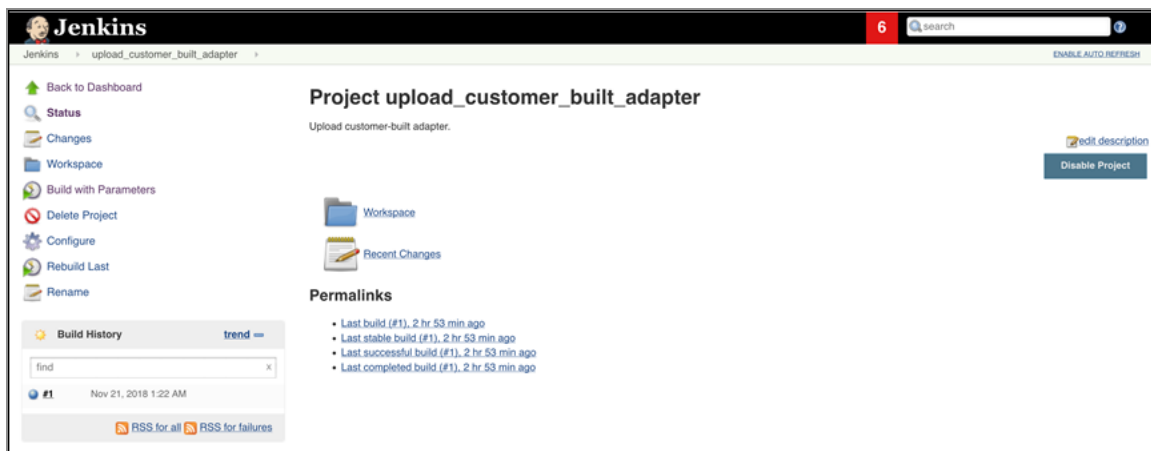
After downloading the .tgz file, follow the steps from here by selecting Boomi-built custom adapter. For more information, see [Prebuilding](#).

For custom adapters developed by Boomi Cloud API Management, make a porting request at [Boomi support](#).

Customer-provided Adapters

This sections describes how to upload adapters built by a customer.

Jenkins job "upload-customer-built-adapter" is for uploading an adapter built by customers, it is accessed using the Jenkins web console:



To upload a customer-built adapter:

- Click "**Browse...**" to select the adapter file.
- Click "**Build**" to upload the adapter to Jenkins server.

Note that:

- A customer-built adapter is built by a customer using the Mashery SDK.
- A customer-built adapter is a zip file.
- The zip file is uploaded to "/var/jenkins_home/userContent/proxy-extension" in the tml-installer container.

When a Docker image for Traffic Manager is built, the customer-built adapter is built into the Docker image. See the following example build output:

```

15:50:14 * Connected to tmgc-installer-host (192.168.4.60) port 8080 (#0)
15:50:14 > GET /userContent/proxy-extension/customer_extension.zip HTTP/1.1
15:50:14 > User-Agent: curl/7.29.0
15:50:14 > Host: tmgc-installer-host:8080
15:50:14 > Accept: */*
15:50:14 >
15:50:14 [0m[91m< HTTP/1.1 200 OK
15:50:14 < Date: Sat, 17 Nov 2018 23:50:14 GMT
15:50:14 < X-Content-Type-Options: nosniff
15:50:14 < Content-Security-Policy: sandbox; default-src 'none'; img-src 'self'; style-src 'self';
15:50:14 < X-WebKit-CSP: sandbox; default-src 'none'; img-src 'self'; style-src 'self';
15:50:14 < X-Content-Security-Policy: sandbox; default-src 'none'; img-src 'self'; style-src 'self';
15:50:14 < Last-Modified: Sat, 17 Nov 2018 23:42:06 GMT
15:50:14 < Expires: Sat, 17 Nov 2018 23:42:06 GMT
15:50:14 < Accept-Ranges: bytes
15:50:14 < Content-Type: application/zip
15:50:14 < Content-Length: 126649
15:50:14 < Server: Jetty(9.4.z-SNAPSHOT)
15:50:14 <
15:50:14 { [data not shown]
15:50:14
100 123k 100 123k 0 0 18.9M 0 --:--:-- --:--:-- --:--:-- 20.1M
15:50:14 * Connection #0 to host tmgc-installer-host left intact
15:50:14 [0mArchive: /tmp/customer_extension.zip
15:50:14 inflating: /tmp/customer-bundles/Base64Adapter-1.0.jar
15:50:14 inflating: /tmp/customer-bundles/Base64UrlAdapter-1.0.jar
15:50:14 inflating: /tmp/customer-bundles/Base64UrlAdapter2-1.0.jar
15:50:14 inflating: /tmp/customer-bundles/lib/codec-1.3.0.v201101211617.jar
15:50:14 inflating: /tmp/customer-bundles/lib/codec-1.4.0.v201209201156.jar
15:50:14 /opt/javaproxy/tools/packager /opt/javaproxy
15:50:15 Buildfile: /opt/javaproxy/tools/packager/build.xml
15:50:15
15:50:15 all:
15:50:15
15:50:15 setUp:
15:50:15 [mkdir] Created dir: /tmp/dist-repository
15:50:15 [mkdir] Created dir: /tmp/dist-repository/source
15:50:15 [mkdir] Created dir: /tmp/dist-repository/source/plugins
15:50:15 [mkdir] Created dir: /tmp/dist-repository/source/features
15:50:15 [mkdir] Created dir: /tmp/dist-repository/destination
15:50:15
15:50:15 packageRepository:
15:50:15 [copy] Copying 2 files to /tmp/dist-repository/source/features
15:50:15
15:50:15 publishRepository:
15:50:15 [zip] Building zip: /tmp/dist-repository/repository.zip

```

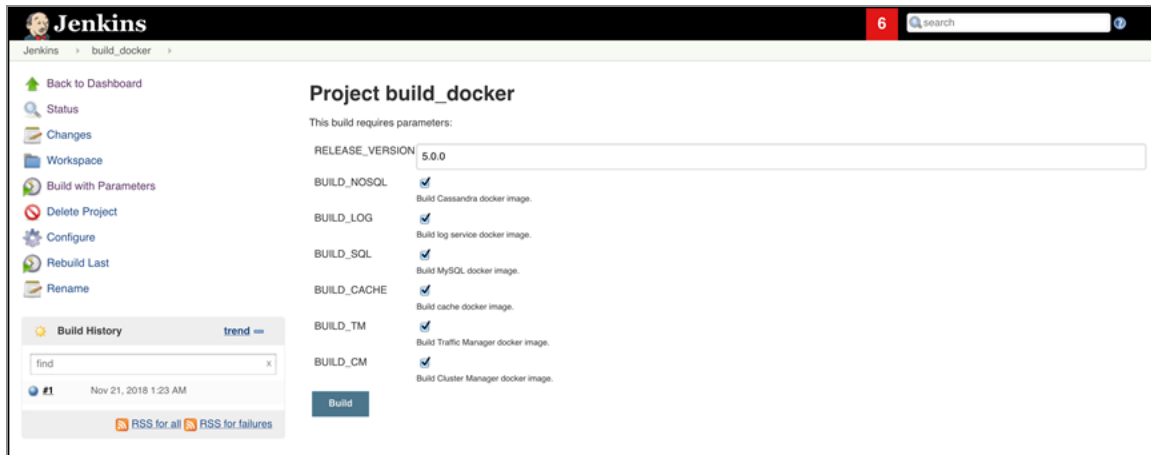
```

15:50:15 BUILD SUCCESSFUL
15:50:15
15:50:15 BUILD SUCCESSFUL
15:50:15 Total time: 0 seconds
15:50:15 /opt/javaproxy
15:50:15 Archive: /tmp/dist-repository/repository.zip
15:50:15  creating: /tmp/dist-install/features/
15:50:15  creating: /tmp/dist-install/plugins/
15:50:15  inflating: /tmp/dist-install/artifacts.xml
15:50:15  inflating: /tmp/dist-install/content.xml
15:50:15  inflating: /tmp/dist-install/features/com.mashery.proxy.local.customer.extensions_
1.0.0.201811172350.jar
15:50:15  inflating: /tmp/dist-install/plugins/com.mashery.customer.Base64Adapter-1.0_
1.0.0.201811172350.jar
15:50:15  inflating: /tmp/dist-install/plugins/com.mashery.customer.Base64UrlAdapter-1.0_
1.0.0.201811172350.jar
15:50:15  inflating: /tmp/dist-install/plugins/com.mashery.customer.Base64UrlAdapter2-1.0_
1.0.0.201811172350.jar
15:50:15 Updating /opt/javaproxy/proxy...
15:50:16 INFO  Profile Local
15:50:16 INFO  Timestamp Sat, 17 Nov 2018 23:50:13 GMT
15:50:16 INFO  Found com.mashery.apiserver.oauth.feature.group 1.16.0.v20181116-0033
Mashery API Server
15:50:16 INFO  Found com.mashery.proxy.local.customer.extensions.feature.group 1.0.0
Customer Extensions
15:50:16 INFO  Found com.mashery.proxy.local.customer.feature.group 1.59.0.v20180813-0728
Customer-specific Bundles
15:50:16 INFO  Found com.mashery.proxy.local.product 1.46.0.v20181116-2143 Mashery Proxy
15:50:16 INFO  Installing com.mashery.proxy.local.customer.extensions.feature.group
1.0.0.201811172350
15:50:17 INFO  Profile updated.
15:50:17 INFO  Timestamp Sat, 17 Nov 2018 23:50:17 GMT
15:50:17 INFO  Found com.mashery.apiserver.oauth.feature.group 1.16.0.v20181116-0033
Mashery API Server
15:50:17 INFO  Found com.mashery.proxy.local.customer.extensions.feature.group
1.0.0.201811172350 Mashery Customer Extensions
15:50:17 INFO  Found com.mashery.proxy.local.customer.feature.group 1.59.0.v20180813-0728
Customer-specific Bundles
15:50:17 INFO  Found com.mashery.proxy.local.product 1.46.0.v20181116-2143 Mashery Proxy
15:50:17 INFO  Operation completed in 1350 ms.

```

Build in Jenkins

Jenkins job "build-docker" for building docker images can be accessed using the Jenkins web console:



Jenkins job "build-docker" can also be started remotely using the Jenkins API as seen in the example below:

```
curl -v -X POST http://docker_host:8080/job/build-docker/buildWithParameters \
--data-urlencode json='{ "RELEASE_VERSION": "5.0.0", \
  "BUILD_NOSQL": "true", \
  "BUILD_LOG": "true", \
  "BUILD_SQL": "true", \
  "BUILD_CACHE": "true", \
  "BUILD_TM": "true", \
  "BUILD_CM": "true" }'
```

After a successful build, the docker images are available. See the following example:

docker images

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
tmgc-cm	v5.0.0.1	68b326911a77	2 hours ago	511MB
tmgc-tm	v5.0.0.1	a93a38c331dd	2 hours ago	630MB
tmgc-cache	v5.0.0.1	20f58738793e	2 hours ago	552MB
tmgc-sql	v5.0.0.1	1091817ad763	2 hours ago	1.5GB
tmgc-log	v5.0.0.1	590d05b637b7	3 hours ago	637MB
tmgc-nosql	v5.0.0.1	b4265e78d765	3 hours ago	536MB

For instructions on how to upload a Customer-built adapter, refer to [Prebuilding](#).

Troubleshooting

High CPU Usage in NoSQL Container

It has been observed that nodes where NoSQL container run are showing high CPU usage on some Linux platforms, despite limited use of Cassandra inside NoSQL per container. Although there are various known reasons of high CPU usage by Cassandra, one possible general cause of high CPU usage on Linux platform is Transparent Hugepages. Many Linux distributions come with Transparent Hugepages enabled by default. When Linux uses Transparent Hugepages, the kernel tries to allocate memory in large chunks (usually 2 MB), rather than 4K. However, some applications such as Cassandra, still allocate memory based on 4K pages. This can cause noticeable performance problems when Linux tries to defrag 2 MB pages.

To confirm this behavior and to reduce the CPU usage, follow the steps below. If the reason is confirmed as described below, you should stop the Local Edition cluster, if possible, as it requires restarting the nodes.

1. SSH in the node.

```
cat /sys/kernel/mm/transparent_hugepage/enabled
```

You should get the following output:

```
[always] madvise never
```

2. If you see this output, open the file `/etc/rc.local` for editing. It will have only one statement (exit 0):

```
sudo vi /etc/rc.local
```

3. Add the following lines before the last statement (exit 0):

```
if test -f /sys/kernel/mm/transparent_hugepage/enabled; then  
echo never > /sys/kernel/mm/transparent_hugepage/enabled
```

```

fi

if test -f /sys/kernel/mm/transparent_hugepage/defrag; then

echo never > /sys/kernel/mm/transparent_hugepage/defrag

fi

```

The file should have the following content after the modification.

```

#!/bin/sh -e
#
# rc.local
#
# This script is executed at the end of each multiuser runlevel.
# Make sure that the script will "exit 0" on success or any other
# value on error.
#
# In order to enable or disable this script just change the execution
# bits.
#
# By default this script does nothing.
if test -f /sys/kernel/mm/transparent_hugepage/enabled; then
    echo never > /sys/kernel/mm/transparent_hugepage/enabled
fi
if test -f /sys/kernel/mm/transparent_hugepage/defrag; then
    echo never > /sys/kernel/mm/transparent_hugepage/defrag
fi
exit 0
~
~
~
~
~
~

```

4. Save and exit.
5. Restart the node. Please perform steps 1-5 on all the nodes.
6. After successful restart, SSH into the node and run the command from step 1.
You should get the following output:

```
[always] madvise never
```

7. Once all nodes are restarted, you can redeploy the API Management - Local Edition cluster.

TML-installer Image Build Failure

If you encounter the following error message while building a new TM image through the tml-installer/jenkins in Local Edition:

Error Message:

```
Cannot retrieve metalink for repository: epel/x86_64. Please verify its path and try again
[0m"yum_install initscripts.x86_64" command filed with exit code 1.
The command '/bin/sh -c /home/builder/docker-build/tm-install.sh' returned a non-zero code: 1
```

Check for the following items:

- The repositories might be temporarily unavailable. Give it some time and try the build again.
- There is a network issue on your machine that is prohibiting connection to the Internet.

If you don't have a dedicated Linux machine to run image build (tml-installer), the following steps are recommended:

1. On a Windows host, create a Linux machine (such as Ubuntu) on a VirtualBox setup.
2. Change the network settings of the Linux machine to Bridged Network.
3. Using Docker for Linux (such as Ubuntu) run tml-installer container and build image job on the Linux machine.

i Note: Currently, Docker for Linux is more stable than Docker for Windows, so Boomi recommends running tml-installer on Docker for Linux.

If issues for creating images still exist, check with your network team for any firewall rules that need to be less restrictive.

Boomi References

Refer to these links to learn more about Boomi privacy policy, terms of service, and Boomi help documentation:

[Privacy Policy](#)

[Terms of Service](#)

[Help Documentation](#)