DCAE MOD Catalog and UI redesign (POC)

DCAE MOD (Microservices Onboarding and Design) is being redesigned with angular based UI to have a great user experience, self managed catalog with APIs to ease access and manage life cycle of different version of microservices.

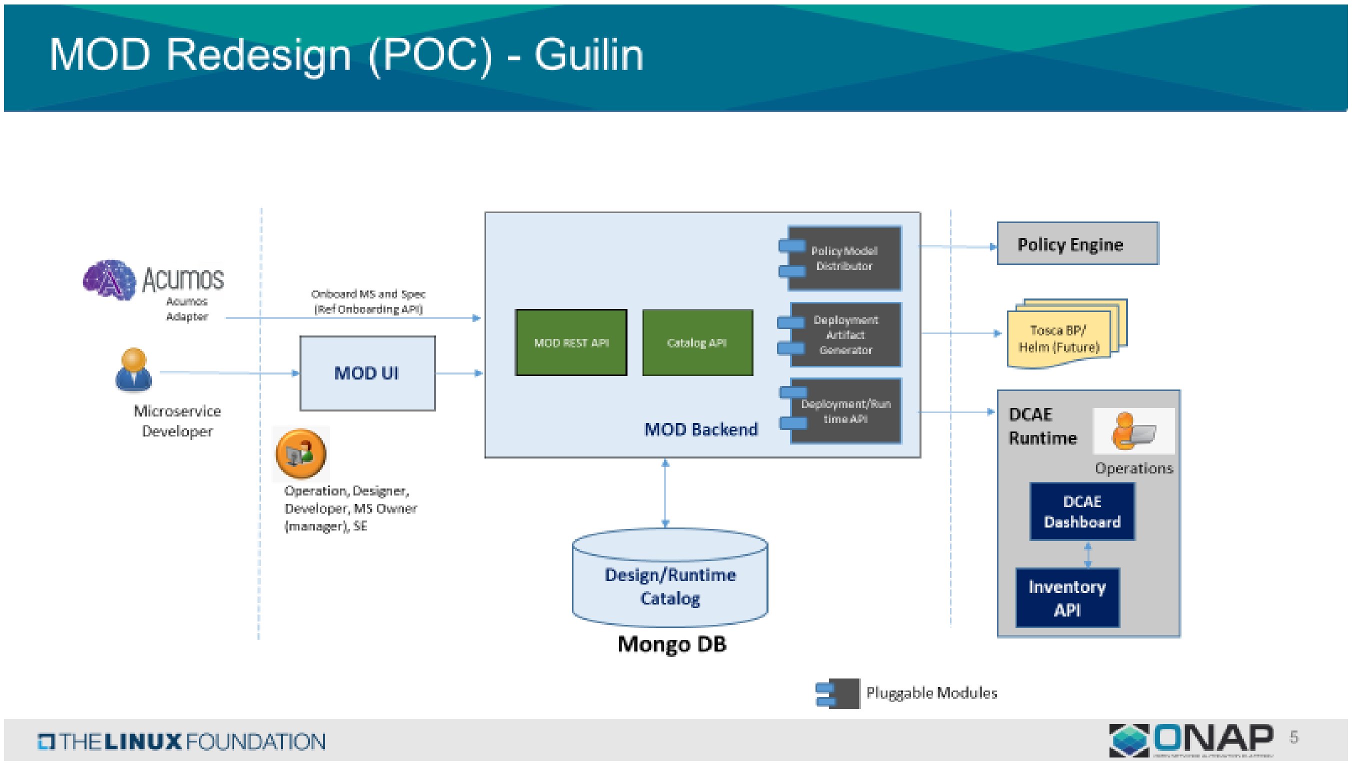
It also provides extensibility to include pluggable modules for generating deployment artifacts and communication with other systems such as Policy Engine and DCAE dashboard.

Details of the target architecture and initial implementation can be found in below ppt slides and initial API Specifications

DCAE MOD Catalog and UI redesign draft - DCAE_MOD_redesign_v3.pptx

API Specification - swagger.yml

Below picture depicts the target architecture of redesigned MOD



It is composed of UI and Backend with Catalog service and Authentication Service.

UI

UI is developed using angular framework and enables users including designers and developers to onboard microservices. It currently provides a user-friendly menu options to onboard microservices and its instances per release, associate a component specification and generate deployment artifact that can be deployed using DCAE dashboard.

It will be enhanced to onboard policy models and data format to support policy-based control and composite microservice models.

Backend

The backend for MOD composed of a catalog service and authentication service. Catalog Service will manage microservices, its instances and generated deployment artifacts. Authentication service will provide access control for designers and developers to perform different operations.

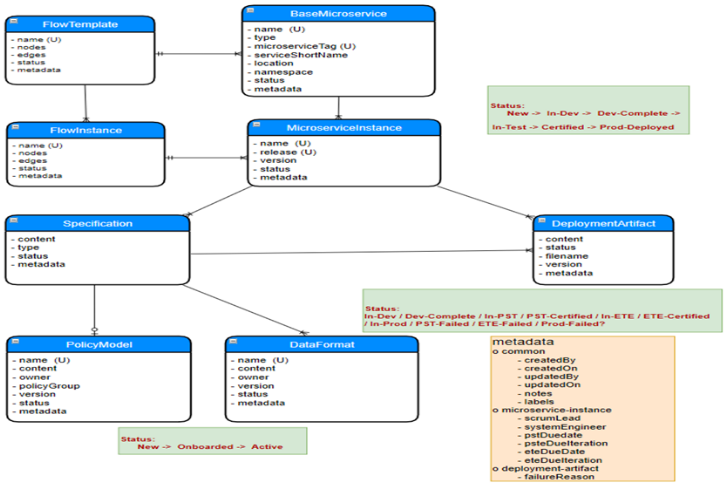
Catalog Service

Catalog service is composed of APIs to manage base microservice, microservice instances, component specifications associated with microservices and to generate deployment artifacts.

Authentication Service

Authentication service provides APIs to manage users, currently used by UI.

Below ER diagram represents the entities present in Mongo Database



User Guide

Deployment using Docker compose

MOD can be deployed using docker-compose with below link for docker-compose,yml file on any Ubuntu VMs

https://git.onap.org/dcaegen2/platform/tree/mod2/assembly/docker-compose.yml

Create a directory and create docker-compose.yml file with the above contents.

Set DACE_HOSTNAME by running export DCAE_HOSTNAME=<IP Address of Server>. This is required for UI to determine the DCAE MOD backend server.

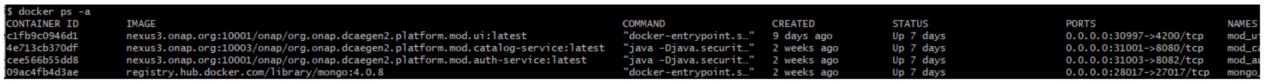
Note : Additional attributes are introduced in Honolulu release are APIs for distributing policies to policy framework and distributing deployment artifacts to DCAE dashboard. If you are planning to use those APIs, export environment variables for POLICYMODEL and DCAE_PLATFORM as specified in above attached docker-compose.yml

Run below command to download the latest built version and deploy on docker container.

```
docker-compose -f docker-compose.yml up &
```

After deployment is successful, you can list using docker ps -a command and it will show the below deployments.

Note: For Guilin POC, the above deployment procedure would be followed to install MOD components. For future releases, it will be enhanced to use helm chart.



```
$ docker ps -a
CONTAINER ID     IMAGE                                                                    COMMAND                  CREATED        STATUS          PORTS                       NAMES
c1fb9c0946d1     nexus3.onap.org:10001/onap/org.onap.dcaegen2.platform.mod.ui:latest               "docker-entrypoint.s…"   9 days ago     Up 7 days       0.0.0.0:30997->4200/tcp     mod_u
4e713cb370df     nexus3.onap.org:10003/onap/org.onap.dcaegen2.platform.mod.catalog-service:latest  "java -Djava.securit…"   2 weeks ago    Up 7 days       0.0.0.0:31001->8080/tcp     mod_c
cee566b55dd8     nexus3.onap.org:10001/onap/org.onap.dcaegen2.platform.mod.auth-service:latest     "java -Djava.securit…"   2 weeks ago    Up 7 days       0.0.0.0:31003->8082/tcp     mod_a
09ac4fb4d3ae     registry.hub.docker.com/library/mongo:4.0.8                              "docker-entrypoint.s…"   2 weeks ago    Up 7 days       0.0.0.0:28017->27017/tcp    mongo_
```

Deployment using Helm (introduced in Honolulu Release)

Download the charts (all the files and directories) from git with below link

https://git.onap.org/dcaegen2/platform/tree/mod2/assembly/helm

Under **components/catalog-service**, update K8SNodeip and PASSWORD for policy framework deployment and DCAE Dashboard deployment and verify other configuration values in **values,yaml**

```
policyModelDevServer: "<K8SNodeip>"

policyModelDevPort: "30522"

policyModelDevUser: "healthcheck"

policyModelDevPassword: "<PASSWORD>"

dcaePlatformDevServer: "<K8SNodeip>"

dcaePlatformDevPort: "30418"
```

```
dcaePlatformDevUser: "su1234"

dcaePlatformDevPassword: "<PASSWORD>"
```

Under components/ui, update K8SNodeip for master node in **values.yaml**

```
dcaeHostname: <K8SNodeip>
```

After updating the above configurations, run below command from the parent directory to generate the charts with all dependencies.

```
helm dep update
```

Then, run below command to install all the components under desired NAMESPACE

```
helm install -n dcaemod2 --namespace <NAMESPACE> ./
```

Verify all the components (pods) are up

```
dcaemod2-mongo-0

dcaemod2-ui

catalog-service

auth-service
```

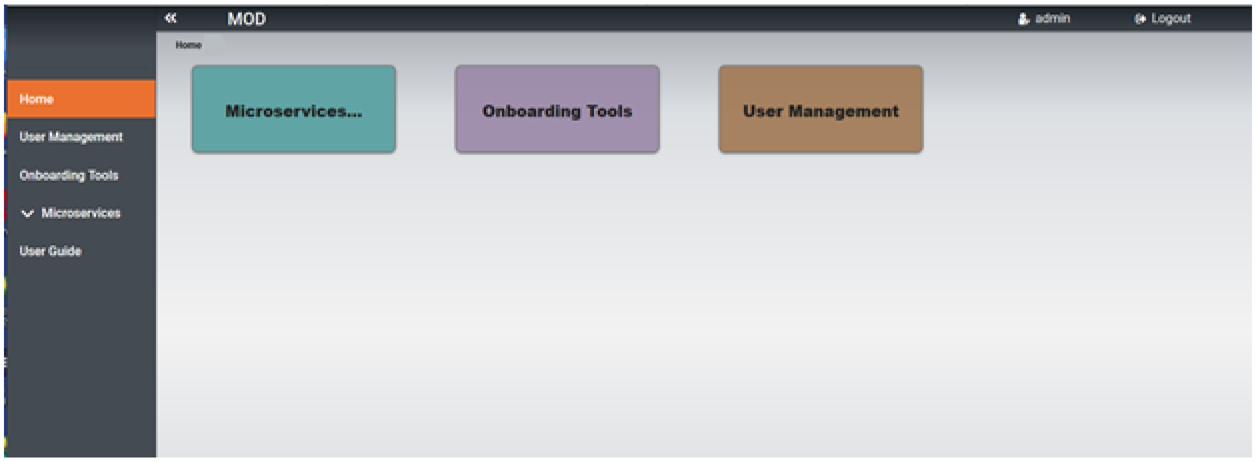
Working with UI

After deploying the mod2 components using docker compose or helm charts, UI will be available in below URL where hostname would be the hostname or IP address of the deployed VM

https:// <hostname>:30997 (or the node port configured in the helm chart - 31009 in the current helm chart)

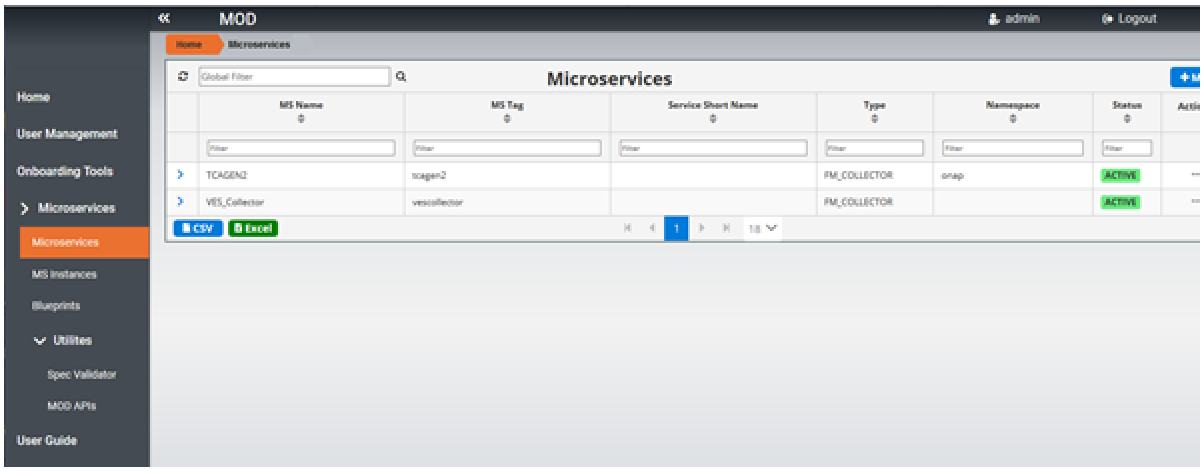
Initial user/password would be set to admin/admin@mod.

Below is the snapshot of initial landing page of MOD UI



Steps to onboard a microservice and generate deployment artifact

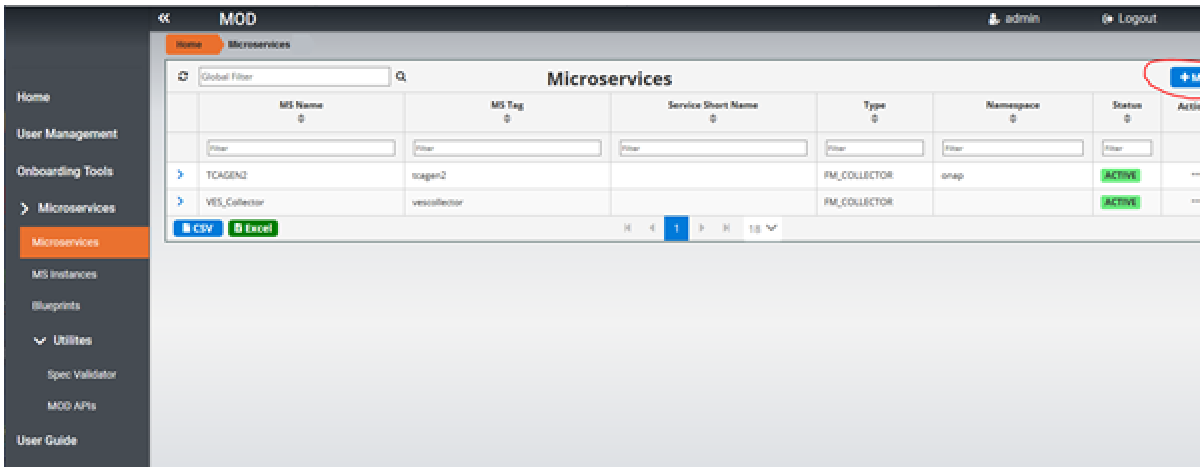
1. Click on Microservices... option on the page, following sample page will be displayed. Currently this page contains on-boarded microservices in the list. For initial deployment, the list will be empty



Microservices

MS Name	MS Tag	Service Short Name	Type	Namespace	Status
TCAGEN2	tcagen2		FM_COLLECTOR	onap	ACTIVE
VES_Collector	vescollector		FM_COLLECTOR		ACTIVE

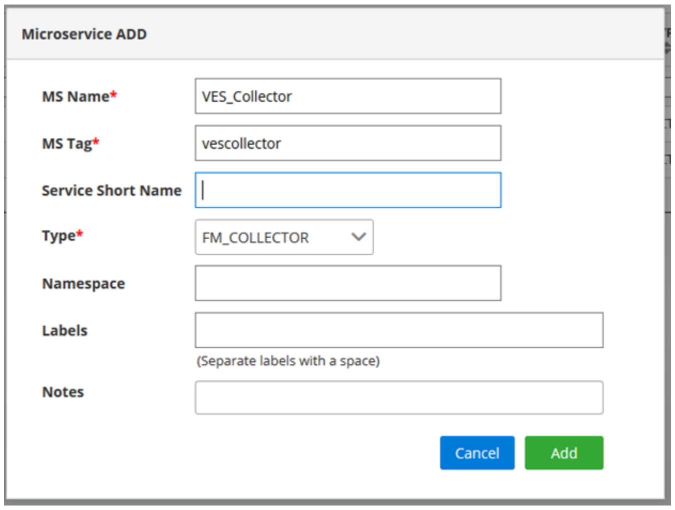
2. Click on +MS button on the page below shown in red to add a Microservice that needs to be onboarded.



Microservices

MS Name	MS Tag	Service Short Name	Type	Namespace	Status
TCAGEN2	tcagen2		FM_COLLECTOR	onap	ACTIVE
VES_Collector	vescollector		FM_COLLECTOR		ACTIVE

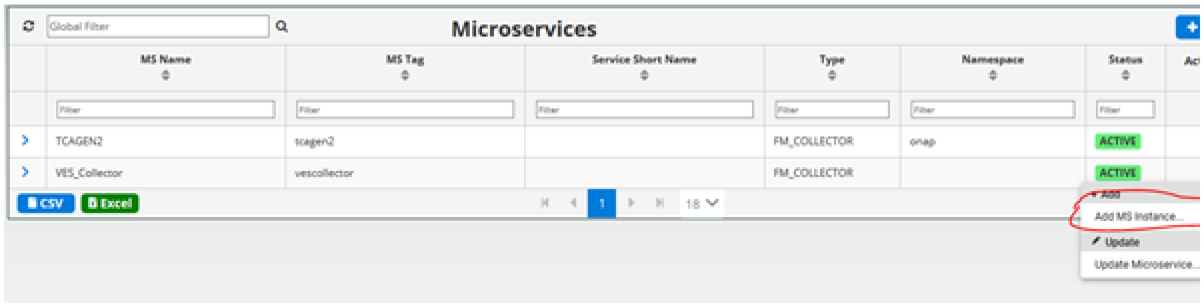
3. Below Popup page will be displayed and enter details of Microservice and press Add to add the Microservce in the list.



Microservice ADD

MS Name*	VES_Collector
MS Tag*	vescollector
Service Short Name	
Type*	FM_COLLECTOR
Namespace	
Labels	(Separate labels with a space)
Notes	

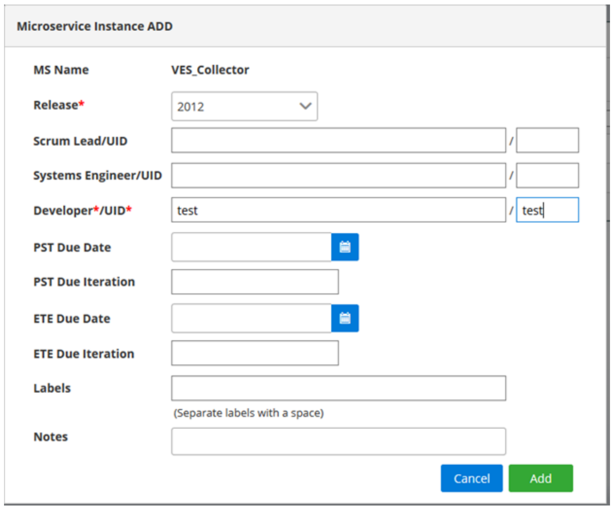
4. To add an instance for a release click on … on the right as circled below in the list of Microservice by selecting s specific Microservice



Microservices

	MS Name	MS Tag	Service Short Name	Type	Namespace	Status
›	TCAGEN2	tcagen2		FM_COLLECTOR	onap	ACTIVE
›	VES_Collector	vescollector		FM_COLLECTOR		ACTIVE

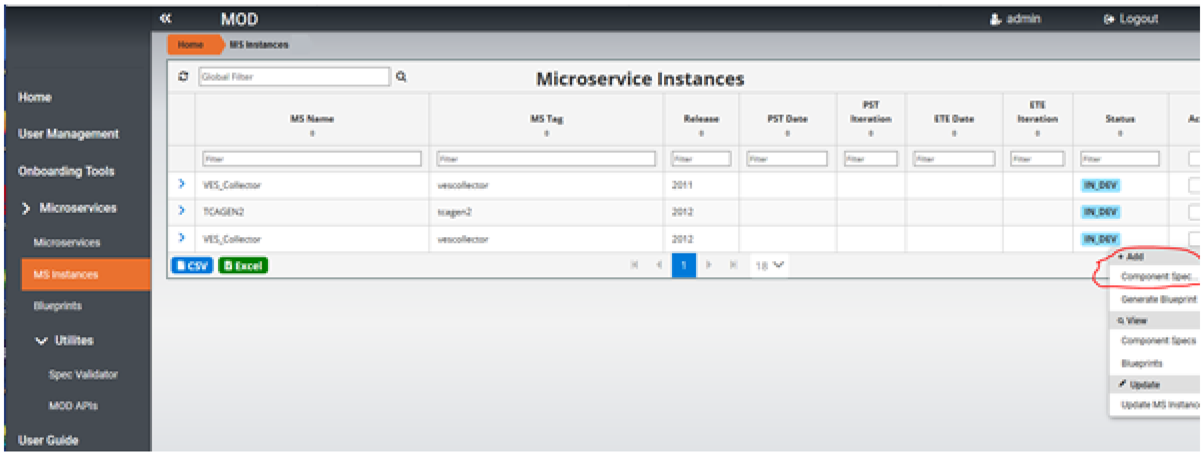
5. Below popup page will be displayed. Select a release and developer details. Then click on Add button to add a Microservice instance.



Microservice Instance ADD

MS Name	VES_Collector
Release*	2012
Scrum Lead/UID	/
Systems Engineer/UID	/
Developer*/UID*	test / test
PST Due Date	
PST Due Iteration	
ETE Due Date	
ETE Due Iteration	
Labels	(Separate labels with a space)
Notes	

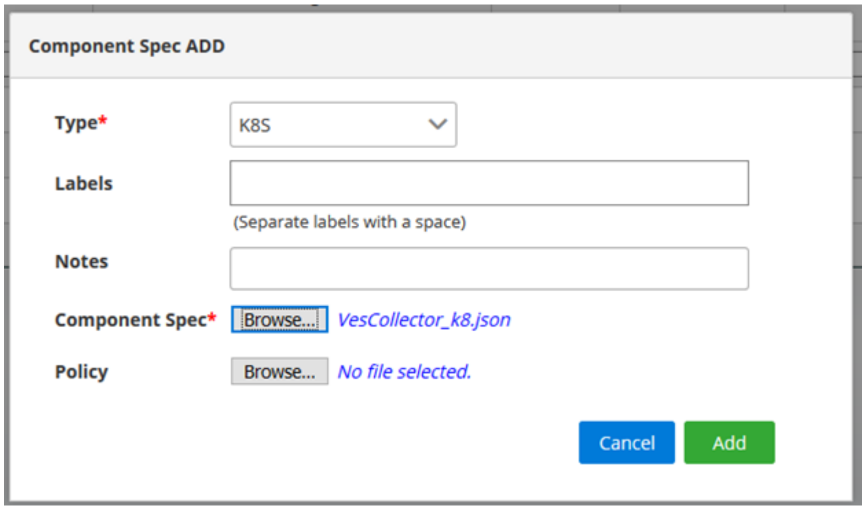
6. To add a component spec for Microservice instance, Select MS Instances on the left menu, it will bring up the list of Microservice instances and choose … on the right side for the specific Microservice instance as below. Click on Add Component spec option.



Microservice Instances

	MS Name	MS Tag	Release	PST Date	PST Iteration	ETE Date	ETE Iteration	Status
›	VES_Collector	vescollector	2011					IN_DEV
›	TCAGEN2	tcagen2	2012					IN_DEV
›	VES_Collector	vescollector	2012					IN_DEV

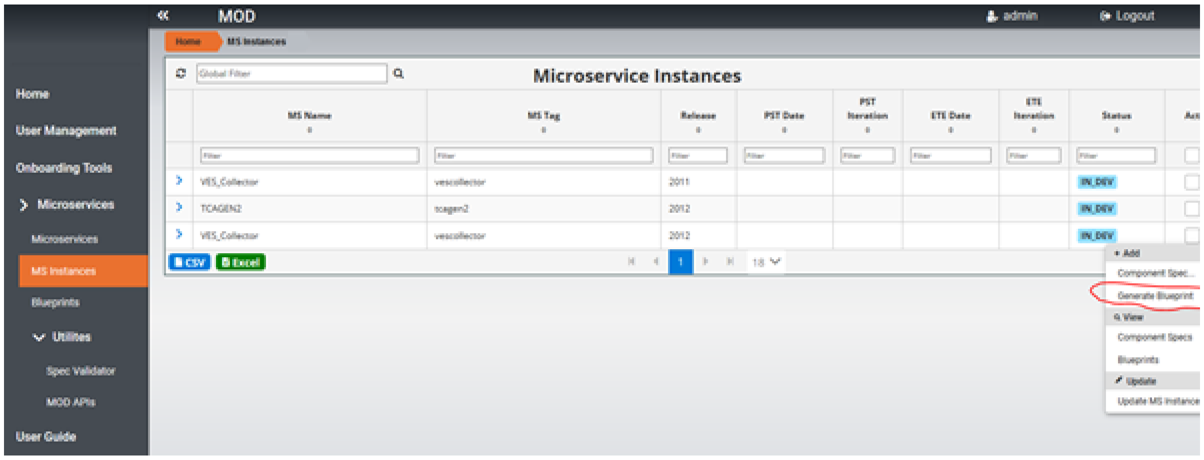
7. Below popup page will be displayed for adding component spec json file. Select the type and browse to choose component spec json file. Choose Add to add the component spec for the specific Microservice instance.



Component Spec ADD

Type*	K8S
Labels	
	(Separate labels with a space)
Notes	
Component Spec*	Browse... *VesCollector_k8.json*
Policy	Browse... *No file selected.*

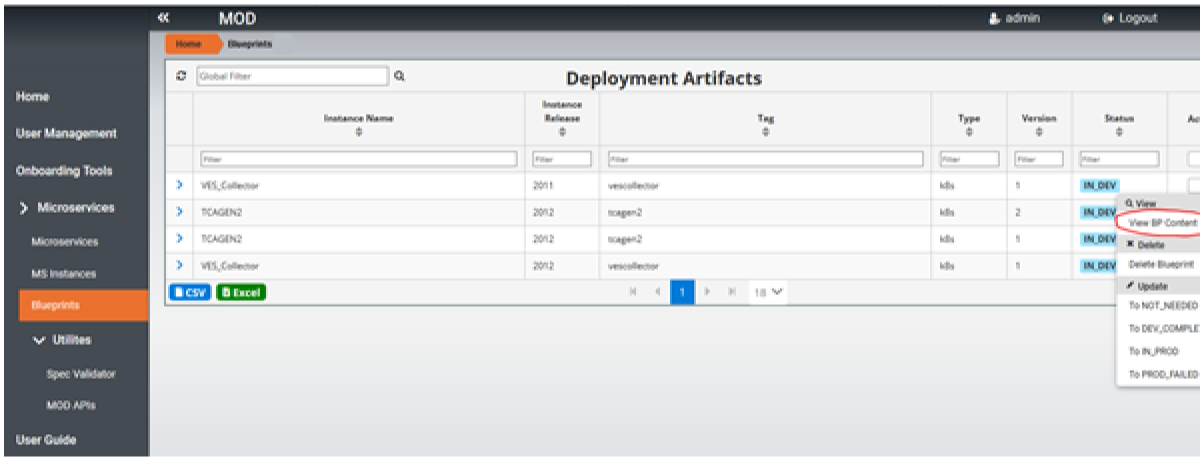
8. To generate deployment artifact, choose … on the right side for the specific Microservice instance as below. Then choose Generate Blueprint option to generate the deployment Artifact.



Microservice Instances

	MS Name	MS Tag	Release	PST Date	PST Iteration	ETE Date	ETE Iteration	Status
›	VES_Collector	vescollector	2011					IN_DEV
›	TCAGEN2	tcagen2	2012					IN_DEV
›	VES_Collector	vescollector	2012					IN_DEV

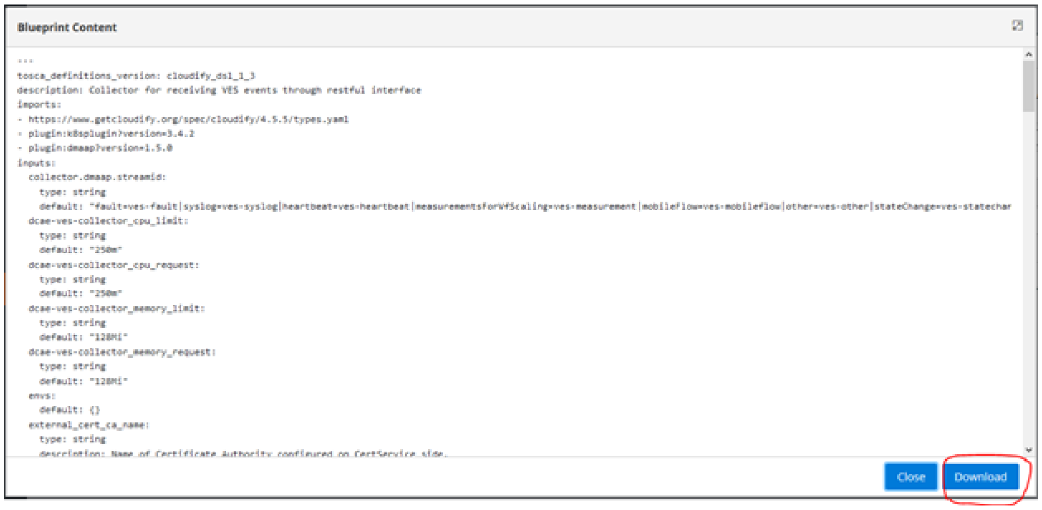
9. Choose Blueprints on left side menu to get a list of blueprints that are generated. Choose … option menu for specific blueprint and choose "View BP Content"



Deployment Artifacts

	Instance Name	Instance Release	Tag	Type	Version	Status
›	VES_Collector	2011	vescollector	k8s	1	IN_DEV
›	TCAGEN2	2012	tcagen2	k8s	2	IN_DEV
›	TCAGEN2	2012	tcagen2	k8s	1	IN_DEV
›	VES_Collector	2012	vescollector	k8s	1	IN_DEV

10. The Blueprint content can be downloaded by choosing download option show below.



```
...
tosca_definitions_version: cloudify_dsl_1_3
description: Collector for receiving VES events through restful interface
imports:
- https://www.getcloudify.org/spec/cloudify/4.5.5/types.yaml
- plugin:k8splugin?version=3.4.2
- plugin:dmaap?version=1.5.0
inputs:
  collector.dmaap.streamid:
    type: string
    default: "fault=ves-fault|syslog=ves-syslog|heartbeat=ves-heartbeat|measurementsForVfScaling=ves-measurement|mobileflow=ves-mobileflow|other=ves-other|stateChange=ves-statechar
  dcae-ves-collector_cpu_limit:
    type: string
    default: "250m"
  dcae-ves-collector_cpu_request:
    type: string
    default: "250m"
  dcae-ves-collector_memory_limit:
    type: string
    default: "128Mi"
  dcae-ves-collector_memory_request:
    type: string
    default: "128Mi"
  envs:
    default: {}
  external_cert_ca_name:
    type: string
    description: Name of Certificate Authority configured on CertService side.
```

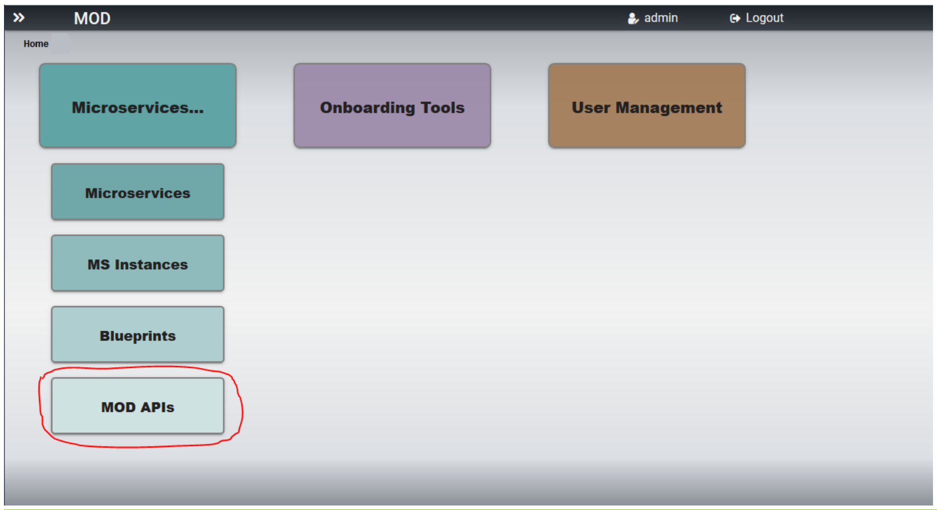
Currently after the deployment artifact is downloaded, you have to go manually to DCAE dashboard to deploy.

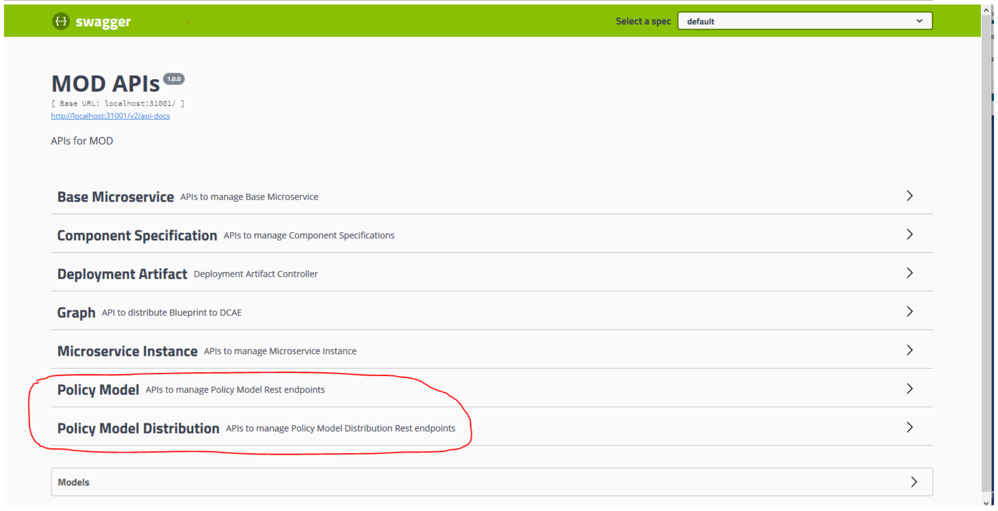
In Honolulu release, below APIs are provided to onboard and distribute policy to policy framwork engine. Also APIs to distribute the deployment artifact on to DCAE Dashboard is available.

On-boarding Policy Model and Distributing to Policy Framework Engine Using API (Introduced in Honolulu Release)

Below APIs are introduced to onboard a policy model and distribute to policy framework engine in catalog service. In order to look into Policy Model APIs, the swagger document is available in UI.

When logged on to UI, select MOD APIs option under UI, it will bring up below swagger UI and will find Policy Model and Policy Model Distribution APIs



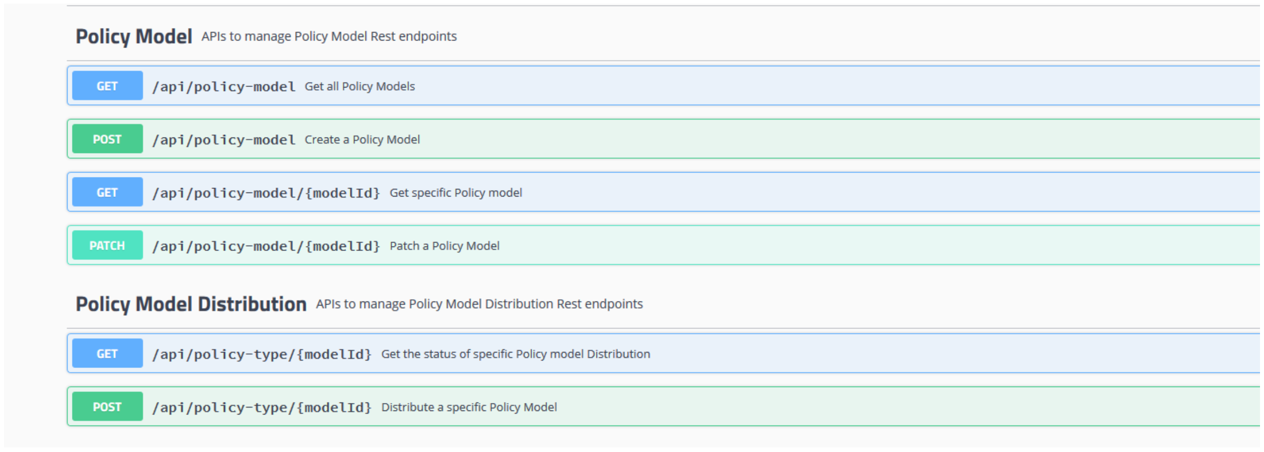


MOD APIs 1.0.0
[Base URL: localhost:31001/]
http://localhost:31001/v2/api-docs

APIs for MOD

Base Microservice APIs to manage Base Microservice
Component Specification APIs to manage Component Specifications
Deployment Artifact Deployment Artifact Controller
Graph API to distribute Blueprint to DCAE
Microservice Instance APIs to manage Microservice Instance
Policy Model APIs to manage Policy Model Rest endpoints
Policy Model Distribution APIs to manage Policy Model Distribution Rest endpoints
Models

Following Policy Model APIs are available in swagger UI and details of the APIs are found when selecting each API



Policy Model APIs to manage Policy Model Rest endpoints

GET /api/policy-model Get all Policy Models
POST /api/policy-model Create a Policy Model
GET /api/policy-model/{modelId} Get specific Policy model
PATCH /api/policy-model/{modelId} Patch a Policy Model

Policy Model Distribution APIs to manage Policy Model Distribution Rest endpoints

GET /api/policy-type/{modelId} Get the status of specific Policy model Distribution
POST /api/policy-type/{modelId} Distribute a specific Policy Model

For on-boarding ahe policy model - http://<Node IP>:31001/api/policy-model?user="admin" and sample input JSON as below.

Sample JSON to create policy model

```
{
   "name": "onap.policies.Monitoring",
   "owner": "admin",
   "version": "1.0.0",
   "content": "tosca_definitions_version: tosca_simple_yaml_1_1_0\r\npolicy_types:\r\n   onap.policies.
Monitoring:\r\n      derived_from: tosca.policies.Root\r\n      version: 1.0.0\r\n      name: onap.policies.
```

```
Monitoring\r\n      description: a base policy type for all policies that govern monitoring provisioning\r\n
onap.policies.monitoring.tcagen2:\r\n      derived_from: onap.policies.Monitoring\r\n      version: 1.0.0
\r\n      name: onap.policies.monitoring.tcagen2\r\n      properties:\r\n         tca.policy:\r\n
type: onap.datatypes.monitoring.tca_policy\r\n            description: TCA Policy JSON\r\n            required:
true\r\ndata_types:\r\n   onap.datatypes.monitoring.metricsPerEventName:\r\n      derived_from: tosca.datatypes.
Root\r\n      properties:\r\n         controlLoopSchemaType:\r\n            type: string\r\n
required: true\r\n            description: Specifies Control Loop Schema Type for the event Name e.g. VNF,
VM\r\n            constraints:\r\n            -  valid_values:\r\n               - VM\r\n               -
VNF\r\n         eventName:\r\n            type: string\r\n            required: true\r\n
description: Event name to which thresholds need to be applied\r\n         policyName:\r\n            type:
string\r\n            required: true\r\n            description: TCA Policy Scope Name\r\n         policyScope:
\r\n            type: string\r\n            required: true\r\n            description: TCA Policy
Scope\r\n         policyVersion:\r\n            type: string\r\n            required: true\r\n
description: TCA Policy Scope Version\r\n         thresholds:\r\n            type: list\r\n
required: true\r\n            description: Thresholds associated with eventName\r\n            entry_schema:
\r\n               type: onap.datatypes.monitoring.thresholds\r\n   onap.datatypes.monitoring.tca_policy:
\r\n      derived_from: tosca.datatypes.Root\r\n      properties:\r\n         domain:\r\n            type:
string\r\n            required: true\r\n            description: Domain name to which TCA needs to be
applied\r\n            default: measurementsForVfScaling\r\n            constraints:\r\n            -  equal:
measurementsForVfScaling\r\n         metricsPerEventName:\r\n            type: list\r\n            required:
true\r\n            description: Contains eventName and threshold details that need to be applied to given
eventName\r\n            entry_schema:\r\n               type: onap.datatypes.monitoring.
metricsPerEventName\r\n   onap.datatypes.monitoring.thresholds:\r\n      derived_from: tosca.datatypes.
Root\r\n      properties:\r\n         closedLoopControlName:\r\n            type: string\r\n
required: true\r\n            description: Closed Loop Control Name associated with the threshold\r\n
closedLoopEventStatus:\r\n            type: string\r\n            required: true\r\n            description:
Closed Loop Event Status of the threshold\r\n            constraints:\r\n            -  valid_values:
\r\n               - ONSET\r\n               - ABATED\r\n         direction:\r\n            type:
string\r\n            required: true\r\n            description: Direction of the threshold\r\n
constraints:\r\n            -  valid_values:\r\n               - LESS\r\n               -
LESS_OR_EQUAL\r\n               - GREATER\r\n               - GREATER_OR_EQUAL\r\n               -
EQUAL\r\n         fieldPath:\r\n            type: string\r\n            required: true\r\n
description: Json field Path as per CEF message which needs to be analyzed for TCA\r\n            constraints:
\r\n            -  valid_values:\r\n               - $.event.measurementsForVfScalingFields.vNicPerformanceArray
[*].receivedTotalPacketsDelta\r\n               - $.event.measurementsForVfScalingFields.vNicPerformanceArray
[*].receivedOctetsDelta\r\n               - $.event.measurementsForVfScalingFields.vNicPerformanceArray[*].
receivedUnicastPacketsDelta\r\n               - $.event.measurementsForVfScalingFields.vNicPerformanceArray[*].
receivedMulticastPacketsDelta\r\n               - $.event.measurementsForVfScalingFields.vNicPerformanceArray
[*].receivedBroadcastPacketsDelta\r\n               - $.event.measurementsForVfScalingFields.
vNicPerformanceArray[*].receivedDiscardedPacketsDelta\r\n               - $.event.
measurementsForVfScalingFields.vNicPerformanceArray[*].receivedErrorPacketsDelta\r\n               - $.event.
measurementsForVfScalingFields.vNicPerformanceArray[*].receivedTotalPacketsAccumulated\r\n               - $.
event.measurementsForVfScalingFields.vNicPerformanceArray[*].receivedOctetsAccumulated\r\n               - $.
event.measurementsForVfScalingFields.vNicPerformanceArray[*].
receivedUnicastPacketsAccumulated\r\n               - $.event.measurementsForVfScalingFields.
vNicPerformanceArray[*].receivedMulticastPacketsAccumulated\r\n               - $.event.
measurementsForVfScalingFields.vNicPerformanceArray[*].receivedBroadcastPacketsAccumulated\r\n               -
$.event.measurementsForVfScalingFields.vNicPerformanceArray[*].
receivedDiscardedPacketsAccumulated\r\n               - $.event.measurementsForVfScalingFields.
vNicPerformanceArray[*].receivedErrorPacketsAccumulated\r\n               - $.event.
measurementsForVfScalingFields.vNicPerformanceArray[*].transmittedTotalPacketsDelta\r\n               - $.event.
measurementsForVfScalingFields.vNicPerformanceArray[*].transmittedOctetsDelta\r\n               - $.event.
measurementsForVfScalingFields.vNicPerformanceArray[*].transmittedUnicastPacketsDelta\r\n               - $.
event.measurementsForVfScalingFields.vNicPerformanceArray[*].transmittedMulticastPacketsDelta\r\n
- $.event.measurementsForVfScalingFields.vNicPerformanceArray[*].
transmittedBroadcastPacketsDelta\r\n               - $.event.measurementsForVfScalingFields.vNicPerformanceArray
[*].transmittedDiscardedPacketsDelta\r\n               - $.event.measurementsForVfScalingFields.
vNicPerformanceArray[*].transmittedErrorPacketsDelta\r\n               - $.event.measurementsForVfScalingFields.
vNicPerformanceArray[*].transmittedTotalPacketsAccumulated\r\n               - $.event.
measurementsForVfScalingFields.vNicPerformanceArray[*].transmittedOctetsAccumulated\r\n               - $.event.
measurementsForVfScalingFields.vNicPerformanceArray[*].transmittedUnicastPacketsAccumulated\r\n               -
$.event.measurementsForVfScalingFields.vNicPerformanceArray[*].
transmittedMulticastPacketsAccumulated\r\n               - $.event.measurementsForVfScalingFields.
vNicPerformanceArray[*].transmittedBroadcastPacketsAccumulated\r\n               - $.event.
measurementsForVfScalingFields.vNicPerformanceArray[*].transmittedDiscardedPacketsAccumulated\r\n
- $.event.measurementsForVfScalingFields.vNicPerformanceArray[*].
transmittedErrorPacketsAccumulated\r\n               - $.event.measurementsForVfScalingFields.cpuUsageArray[*].
cpuIdle\r\n               - $.event.measurementsForVfScalingFields.cpuUsageArray[*].
cpuUsageInterrupt\r\n               - $.event.measurementsForVfScalingFields.cpuUsageArray[*].
cpuUsageNice\r\n               - $.event.measurementsForVfScalingFields.cpuUsageArray[*].
cpuUsageSoftIrq\r\n               - $.event.measurementsForVfScalingFields.cpuUsageArray[*].
```

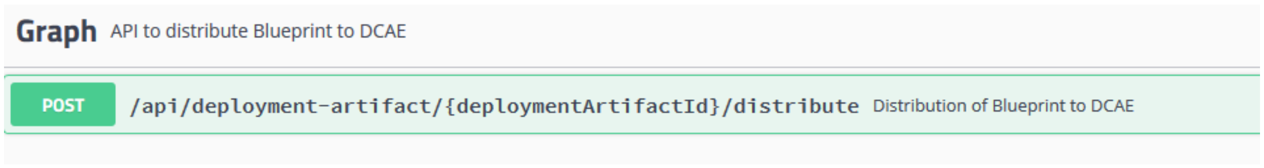
```
cpuUsageSteal\r\n                - $.event.measurementsForVfScalingFields.cpuUsageArray[*].
cpuUsageSystem\r\n                - $.event.measurementsForVfScalingFields.cpuUsageArray[*].
cpuWait\r\n                - $.event.measurementsForVfScalingFields.cpuUsageArray[*].
percentUsage\r\n                - $.event.measurementsForVfScalingFields.meanRequestLatency\r\n                -
$.event.measurementsForVfScalingFields.memoryUsageArray[*].memoryBuffered\r\n                - $.event.
measurementsForVfScalingFields.memoryUsageArray[*].memoryCached\r\n                - $.event.
measurementsForVfScalingFields.memoryUsageArray[*].memoryConfigured\r\n                - $.event.
measurementsForVfScalingFields.memoryUsageArray[*].memoryFree\r\n                - $.event.
measurementsForVfScalingFields.memoryUsageArray[*].memoryUsed\r\n                - $.event.
measurementsForVfScalingFields.additionalMeasurements[*].arrayOfFields[0].value\r\n         severity:
\r\n            type: string\r\n            required: true\r\n            description: Threshold Event
Severity\r\n            constraints:\r\n            -  valid_values:\r\n                -
CRITICAL\r\n                - MAJOR\r\n                - MINOR\r\n                - WARNING\r\n                -
NORMAL\r\n         thresholdValue:\r\n            type: integer\r\n            required: true\r\n
description: Threshold value for the field Path inside CEF message\r\n         version:\r\n            type:
string\r\n            required: true\r\n            description: Version number associated with the
threshold"
}
```

The sample response will contain "id": "604bed36cbb899758eb2734d", which can be used in get, update, and distribute a policy model to policy framework engine.

Distributing Deployment Artifact to DCAE Dashboard Using API (Introduced in Honolulu Release)

Below APIs are introduced to distribute the generated deployment artifact (blueprint) to DCAE Dashboard. Once distributed, it will be available in DCAE dashboard for deployment.

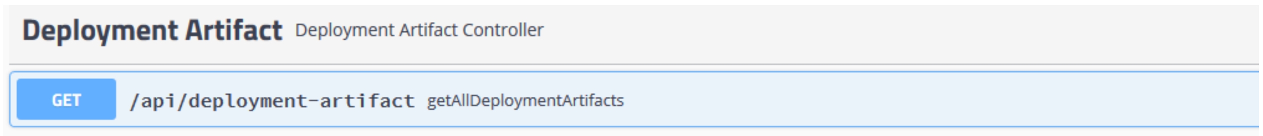
Under MOD APIs swagger UI, you will find this API as below.



Graph API to distribute Blueprint to DCAE

POST /api/deployment-artifact/{deploymentArtifactId}/distribute Distribution of Blueprint to DCAE

Deployment Artifact ID can retrieved by calling get all deployment artifacts using below API



Deployment Artifact Deployment Artifact Controller

GET /api/deployment-artifact getAllDeploymentArtifacts

API Reference : swagger.yml - swagger.yml

Note: Update in progress...