# CAT Customization using l3vpn package

[CAT Customization using l3vpn package 1](#_Toc100155336)

[Develop l3vpn package for NSO 3](#_Toc100155337)

[Get L3vpn demo package from NSO examples 3](#_Toc100155338)

[Modifications for this session 3](#_Toc100155339)

[Develop external notifications package for NSO 4](#_Toc100155340)

[Download CNC SDK , setup and create project 4](#_Toc100155341)

[Review CNC3.0 steps at internal documentation site at: 4](#_Toc100155342)

[Download sdk 4](#_Toc100155343)

[Verify Java JDK11 is installed 5](#_Toc100155344)

[Installing JDK11 5](#_Toc100155345)

[Setup Crosswork Active Topology customization environment 5](#_Toc100155346)

[Create a Project 6](#_Toc100155347)

[Crosswork Active Topology inventory plugin 7](#_Toc100155348)

[Copy yang file from NSO to here 7](#_Toc100155349)

[Find the java files to edit 7](#_Toc100155350)

[Edit the first file MyprojectRestconfDataReader.java to set yangpaths 8](#_Toc100155351)

[Edit the 2nd file MyprojectServiceInventoryPlugin.java 9](#_Toc100155352)

[Final file versions 10](#_Toc100155353)

[Build inventory 12](#_Toc100155354)

[Edit myproject-tsdn-fp-package service-meta-data.json 13](#_Toc100155355)

[Find the file 13](#_Toc100155356)

[Make edits to match the NSO yang file 13](#_Toc100155357)

[Final diffs to review 14](#_Toc100155358)

[CNC UI Provisioning View 14](#_Toc100155359)

[Service Overlay Plugin Configuration 15](#_Toc100155360)

[Find files to edit 15](#_Toc100155361)

[Service JSON from NSO 16](#_Toc100155362)

[Overlay Model to populate 17](#_Toc100155363)

[How to find values in this example using jsonpath 19](#_Toc100155364)

[Edit MyprojectParser.java 20](#_Toc100155365)

[Build complete CNC CAT package 23](#_Toc100155366)

[Upload and Deploy CNC CAT package 23](#_Toc100155367)

[Getting Ticket 24](#_Toc100155368)

[Get Token 24](#_Toc100155369)

[Upload and deploy new CAT function pack. 24](#_Toc100155370)

[Verify on CNC UI 24](#_Toc100155371)

## Develop l3vpn package for NSO

### Get L3vpn demo package from NSO examples

This demo uses l3vpn package from examples.ncs/service-provider/mpls-vpn/packages/l3vpn .

### Modifications for this session

Below modifications done to l3vpn package :

1. Add Organization and revision date

organization "TME";

revision 2021-07-18 {

description

"initial revision";

}

2. Add identity “ l3vpn-endpoint" and l3vpn-plan under container vpn. This is needed to be a sibling in CNC3.0, and in CNC4.0 can be outside of the service

identity l3vpn-endpoint {

base ncs:plan-component-type;

}

{

list l3vpn-plan {

config false;

tailf:cdb-oper {

tailf:persistent true;

}

key "name";

leaf name {

type string;

}

uses ncs:plan-data;

}

3. Add “pe-device” leaf under list endpoint to simplify this excercise

leaf pe-device {

mandatory true;

type leafref {

path "/ncs:devices/ncs:device/ncs:name";

}

}

4. Add leaf “vrf export-policy” to l3vpn to enable On Demand Nexthop mapping to SR-TE policies

leaf vrf-export-policy {

description "Export policy for this VPN" ;

type string;

}

Modified l3vpn package available in Git Repo : - <https://wwwin-github.cisco.com/spna-tme/CNC-CAT-Customization-Example/tree/master/demo/l3vpn>

Create L3VPN Service based on payloads provided in Git : <https://wwwin-github.cisco.com/spna-tme/CNC-CAT-Customization-Example/tree/master/demo>

1. Load merge topology.xml
2. Load merge vpn.xml

Perform commit to ensure service is created successfully.

## Develop external notifications package for NSO

1. Load External Plan Package provided in Git : - <https://wwwin-github.cisco.com/spna-tme/CNC-CAT-Customization-Example/tree/master/demo>
2. Create Bootstrap on NSO by loading config in Git - <https://wwwin-github.cisco.com/spna-tme/CNC-CAT-Customization-Example/blob/master/External-Plan-PoC/bootstrap.xml>

Load merge bootstrap.xml and commit to load config on NSO.

3. Perform redeploy of l3vpn ford service

admin@ncs(config)# vpn l3vpn ford re-deploy

## Download CNC SDK , setup and create project

### Review CNC3.0 steps at internal documentation site at:

<https://wiki.cisco.com/display/ROBOTDEV/CNC+3.0+Service+Extensibility+Guide>

### Download sdk

[tbk@vms-tme-centos2 ~]$ mkdir cat-customization-doc

[tbk@vms-tme-centos2 ~]$

[tbk@vms-tme-centos2 ~]$

[tbk@vms-tme-centos2 ~]$ cd cat-customization-doc/

[tbk@vms-tme-centos2 cat-customization-doc]$ wget https://engci-maven.cisco.com/artifactory/robot-dev-group/com/cisco/sp/cw/tsdn/tools/tsdn-fp-sdk/3.0.0/tsdn-fp-sdk-3.0.0-bin.zip

--2022-03-30 17:28:38-- https://engci-maven.cisco.com/artifactory/robot-dev-group/com/cisco/sp/cw/tsdn/tools/tsdn-fp-sdk/3.0.0/tsdn-fp-sdk-3.0.0-bin.zip

Resolving engci-maven.cisco.com (engci-maven.cisco.com)... 64.102.211.28, 10.83.106.29

Connecting to engci-maven.cisco.com (engci-maven.cisco.com)|64.102.211.28|:443... connected.

HTTP request sent, awaiting response... 200 OK

Length: 14995570 (14M) [application/zip]

Saving to: ‘tsdn-fp-sdk-3.0.0-bin.zip’

100%[======================================================================================================>] 14,995,570 87.4MB/s in 0.2s

2022-03-30 17:28:38 (87.4 MB/s) - ‘tsdn-fp-sdk-3.0.0-bin.zip’ saved [14995570/14995570]

[tbk@vms-tme-centos2 cat-customization-doc]$

[tbk@vms-tme-centos2 cat-customization-doc]$ unzip tsdn-fp-sdk-3.0.0-bin.zip

[tbk@vms-tme-centos2 cat-customization-doc]$ ls -rlt

total 14652

drwxr-xr-x 9 tbk tbk 4096 Nov 16 17:17 tsdn-fp-sdk

-rw-rw-r-- 1 tbk tbk 14995570 Nov 16 20:18 tsdn-fp-sdk-3.0.0-bin.zip

[tbk@vms-tme-centos2 cat-customization-doc]$

### Verify Java JDK11 is installed

[tbk@vms-tme-centos2 cat-customization-doc]$ java --version

openjdk 11.0.14.1 2022-02-08 LTS

OpenJDK Runtime Environment Corretto-11.0.14.10.1 (build 11.0.14.1+10-LTS)

OpenJDK 64-Bit Server VM Corretto-11.0.14.10.1 (build 11.0.14.1+10-LTS, mixed mode)

[tbk@vms-tme-centos2 cat-customization-doc]$

### Installing JDK11

See instructions here: <https://docs.aws.amazon.com/corretto/latest/corretto-11-ug/generic-linux-install.html>

wget https://corretto.aws/downloads/latest/amazon-corretto-11-x64-linux-**jdk**.rpm

sudo rpm -i amazon-corretto-11-x64-linux-**jdk**.rpm

### Setup Crosswork Active Topology customization environment

Run Setup once to create the sdkrc file, this will download artifacts and may take 20-30 minutes for the first time.

[tbk@vms-tme-centos2 bin]$ pwd

/home/tbk/cat-customization-doc/tsdn-fp-sdk/bin

[tbk@vms-tme-centos2 bin]$ ./setup

setting up the tsdn tp sdk environment...

**Apache Maven 3.5.4 (1edded0938998edf8bf061f1ceb3cfdeccf443fe; 2018-06-17T14:33:14-04:00)**

Maven home: /home/tbk/cat-customization-doc/tsdn-fp-sdk/mvn-tools/mvn/apache-maven-3.5.4

Java version: 11.0.14.1, vendor: Amazon.com Inc., runtime: /usr/lib/jvm/java-11-amazon-corretto

Default locale: en\_US, platform encoding: UTF-8

OS name: "linux", version: "3.10.0-693.21.1.el7.x86\_64", arch: "amd64", family: "unix"

Generate /home/tbk/cat-customization-doc/tsdn-fp-sdk/sdkrc

Installiing TSDN FP Dependencies to local maven repository ....

Installing TSDN FP Library dependencies...

[**INFO**] Scanning for projects...

Downloading from opendaylight-mirror: https://nexus.opendaylight.org/content/repositories/public/org/apache/maven/plugins/maven-clean-plugin/2.5/maven-clean-plugin-2.5.pom

A sdkrc file is now created in the base directory

[tbk@vms-tme-centos2 tsdn-fp-sdk]$ pwd

/home/tbk/cat-customization-doc/tsdn-fp-sdk

[tbk@vms-tme-centos2 tsdn-fp-sdk]$ ls -rlt

total 64

-rw-rw-r-- 1 tbk tbk 16871 Nov 16 16:04 README.md

-rw-rw-r-- 1 tbk tbk 179 Nov 16 17:17 build-info.properties

-rw-rw-r-- 1 tbk tbk 18096 Nov 16 17:17 README.html

drwxr-xr-x 2 tbk tbk 4096 Nov 16 17:17 swagger

drwxr-xr-x 2 tbk tbk 60 Nov 16 17:17 postman

drwxr-xr-x 2 tbk tbk 4096 Nov 16 17:17 lib

drwxr-xr-x 4 tbk tbk 32 Nov 16 17:17 examples

drwxr-xr-x 3 tbk tbk 20 Nov 16 17:17 docs

drwxr-xr-x 2 tbk tbk 4096 Nov 16 17:17 bin

-rw-rw-r-- 1 tbk tbk 618 Mar 24 15:43 sdkrc

drwxr-xr-x 4 tbk tbk 4096 Mar 24 15:43 mvn-tools

drwxrwxr-x 5 tbk tbk 129 Mar 24 16:24 tbk-test-tsdn-fp

[tbk@vms-tme-centos2 tsdn-fp-sdk]$ cat sdkrc

##

## tsdn sdk environment - source this file (./sdkrc) to setup the tsdn sdk environment

##

TSDN\_FP\_SDK\_HOME=/home/tbk/cat-customization-doc/tsdn-fp-sdk

MAVEN\_HOME=/home/tbk/cat-customization-doc/tsdn-fp-sdk/mvn-tools/mvn/apache-maven-3.5.4

MVN\_SETTINGS=$TSDN\_FP\_SDK\_HOME/mvn-tools/mvn/mvn-settings.xml

# local maven repository location

MVN\_REPO\_LOCAL=/home/tbk/cat-customization-doc/tsdn-fp-sdk/mvn-tools/.m2/repository

PATH=$TSDN\_FP\_SDK\_HOME/bin:$MAVEN\_HOME/bin:$PATH

export TSDN\_FP\_SDK\_HOME MAVEN\_HOME MVN\_SETTINGS MVN\_REPO\_LOCAL PATH

alias tsdn\_mvn='mvn -s $MVN\_SETTINGS -Dmaven.repo.local=$MVN\_REPO\_LOCAL '

[tbk@vms-tme-centos2 tsdn-fp-sdk]$

Source this sdkrc file into your shell to set environment variables

Source sdkrc

[tbk@vms-tme-centos2 tsdn-fp-sdk]$ pwd

/home/tbk/cat-customization-doc/tsdn-fp-sdk

[tbk@vms-tme-centos2 tsdn-fp-sdk]$ source sdkrc

[tbk@vms-tme-centos2 tsdn-fp-sdk]$

### Create a Project

[tbk@vms-tme-centos2 tsdn-fp-sdk]$ pwd

/home/tbk/cat-customization-doc/tsdn-fp-sdk

[tbk@vms-tme-centos2 tsdn-fp-sdk]$ create-prj myproject

creating tsdn tp maven project(s)...

[INFO] Scanning for projects...

[INFO] Downloading from opendaylight-mirror: <https://nexus.opendaylight.org/content/repositories/public/org/apache/maven/plugins/maven-deploy-plugin/2.7/maven-deploy-plugin-2.7.pom>

.. more downloads for 20-30 minutes

At the end of it there should be 3 directories created

[INFO] Project created from Archetype in dir: /home/tbk/cat-customization-doc/tsdn-fp-sdk/myproject-tsdn-fp

[INFO] ------------------------------------------------------------------------

[INFO] BUILD SUCCESS

[INFO] ------------------------------------------------------------------------

[INFO] Total time: 19:27 min

[INFO] Finished at: 2022-03-30T18:03:17-04:00

[INFO] ------------------------------------------------------------------------

[tbk@vms-tme-centos2 tsdn-fp-sdk]$

[tbk@vms-tme-centos2 tsdn-fp-sdk]$

[tbk@vms-tme-centos2 tsdn-fp-sdk]$ pwd

/home/tbk/cat-customization-doc/tsdn-fp-sdk

[tbk@vms-tme-centos2 tsdn-fp-sdk]$ ls -lrt

total 64

-rw-rw-r-- 1 tbk tbk 16871 Nov 16 16:04 README.md

-rw-rw-r-- 1 tbk tbk 179 Nov 16 17:17 build-info.properties

-rw-rw-r-- 1 tbk tbk 18096 Nov 16 17:17 README.html

drwxr-xr-x 2 tbk tbk 4096 Nov 16 17:17 swagger

drwxr-xr-x 2 tbk tbk 60 Nov 16 17:17 postman

drwxr-xr-x 2 tbk tbk 4096 Nov 16 17:17 lib

drwxr-xr-x 4 tbk tbk 32 Nov 16 17:17 examples

drwxr-xr-x 3 tbk tbk 20 Nov 16 17:17 docs

drwxr-xr-x 2 tbk tbk 4096 Nov 16 17:17 bin

-rw-rw-r-- 1 tbk tbk 630 Mar 30 17:35 sdkrc

drwxr-xr-x 4 tbk tbk 4096 Mar 30 17:35 mvn-tools

drwxrwxr-x 5 tbk tbk 132 Mar 30 18:03 myproject-tsdn-fp

[tbk@vms-tme-centos2 tsdn-fp-sdk]$ cd myproject-tsdn-fp/

[tbk@vms-tme-centos2 myproject-tsdn-fp]$ ls -lrt

total 4

-rw-rw-r-- 1 tbk tbk 1204 Mar 30 18:03 pom.xml

drwxrwxr-x 3 tbk tbk 30 Mar 30 18:03 myproject-tsdn-fp-inventory-plugin

drwxrwxr-x 3 tbk tbk 30 Mar 30 18:03 myproject-tsdn-fp-overlay-plugin

drwxrwxr-x 3 tbk tbk 30 Mar 30 18:03 myproject-tsdn-fp-package

[tbk@vms-tme-centos2 myproject-tsdn-fp]$

## Crosswork Active Topology inventory plugin

### Copy yang file from NSO to here

[tbk@vms-tme-centos2 myproject-tsdn-fp]$ cd myproject-tsdn-fp-inventory-plugin/

[tbk@vms-tme-centos2 myproject-tsdn-fp-inventory-plugin]$ ls

pom.xml src

[tbk@vms-tme-centos2 myproject-tsdn-fp-inventory-plugin]$ cd src/main/yang/

[tbk@vms-tme-centos2 yang]$

[tbk@vms-tme-centos2 yang]$ ls -lrt

total 4

-rw-rw-r-- 1 tbk tbk 896 Mar 30 18:03 myproject.yang

[tbk@vms-tme-centos2 yang]$ rm myproject.yang

[tbk@vms-tme-centos2 yang]$ cp ~/l3vpn.yang .

[tbk@vms-tme-centos2 yang]$ ls -lrt

total 8

-rw-r--r-- 1 tbk tbk 7318 Mar 30 18:10 l3vpn.yang

[tbk@vms-tme-centos2 yang]$

### Find the java files to edit

cd $TSDN\_FP\_SDK\_HOME/mypro\*

[tbk@vms-tme-centos2 myproject-tsdn-fp]$ ls -lrt

total 4

-rw-rw-r-- 1 tbk tbk 1204 Mar 30 18:03 pom.xml

drwxrwxr-x 3 tbk tbk 30 Mar 30 18:03 myproject-tsdn-fp-inventory-plugin

drwxrwxr-x 3 tbk tbk 30 Mar 30 18:03 myproject-tsdn-fp-overlay-plugin

drwxrwxr-x 3 tbk tbk 30 Mar 30 18:03 myproject-tsdn-fp-package

[tbk@vms-tme-centos2 myproject-tsdn-fp]$ cd myproject-tsdn-fp-inventory-plugin/

[tbk@vms-tme-centos2 myproject-tsdn-fp-inventory-plugin]$

[tbk@vms-tme-centos2 myproject-tsdn-fp-inventory-plugin]$ find . -name '\*.java'

./src/main/java/com/cisco/cw/tsdn/fp/myproject/inventory/MyprojectRestconfDataReader.java

./src/main/java/com/cisco/cw/tsdn/fp/myproject/inventory/MyprojectServiceInventoryPlugin.java

./src/test/java/com/cisco/cw/tsdn/fp/myproject/inventory/MyprojectTest.java

[tbk@vms-tme-centos2 myproject-tsdn-fp-inventory-plugin]$

[tbk@vms-tme-centos2 myproject-tsdn-fp-inventory-plugin]$

[tbk@vms-tme-centos2 myproject-tsdn-fp-inventory-plugin]$

[tbk@vms-tme-centos2 myproject-tsdn-fp-inventory-plugin]$

### Edit the first file MyprojectRestconfDataReader.java to set yangpaths

[tbk@vms-tme-centos2 myproject-tsdn-fp-inventory-plugin]$ grep -i yangpath ./src/main/java/com/cisco/cw/tsdn/fp/myproject/inventory/MyprojectRestconfDataReader.java

import com.cisco.sp.cw.cat.nbi.restconf.fw.store.**YangPath**;

@**YangPath**(value = "(http://cisco.com/ns/cw/tsdn/service/myproject?revision=2022-03-30)myproject", type = LogicalDatastoreType.OPERATIONAL)

@**YangPath**(value = "(http://cisco.com/ns/cw/tsdn/service/myproject?revision=2022-03-30)myproject-plan", type = LogicalDatastoreType.OPERATIONAL)

[tbk@vms-tme-centos2 myproject-tsdn-fp-inventory-plugin]$

[tbk@vms-tme-centos2 myproject-tsdn-fp-inventory-plugin]$

Use any editor and set the yangpath for the above two using the following rules. See below for hints on how to find those in the l3vpn.yang used in this example.

Yangpath for service is "(<namespace>?<revision>)<containerpath>/<servicelistname>"

Yangpath for service plan-data is "(<namespace>?<revision>)<containerpath>/<planlistname>"

The results after editing should look like this in our example.

@**YangPath**(value = "(http://com/example/l3vpn?revision=2021-07-18)vpn/l3vpn", type = LogicalDatastoreType.OPERATIONAL)

@**YangPath**(value = "(http://com/example/l3vpn?revision=2021-07-18)vpn/l3vpn-plan", type = LogicalDatastoreType.OPERATIONAL)

How to find namespace in yang module:

[tbk@vms-tme-centos2 myproject-tsdn-fp-inventory-plugin]$ grep namespace src/main/yang/l3vpn.yang

**namespace** "http://com/example/l3vpn";

How to find revision in yang module:

[tbk@vms-tme-centos2 myproject-tsdn-fp-inventory-plugin]$ grep revision src/main/yang/l3vpn.yang

**revision** 2021-07-18 {

"initial **revision**";

[tbk@vms-tme-centos2 myproject-tsdn-fp-inventory-plugin]$

How to find module name in yang module:

[tbk@vms-tme-centos2 myproject-tsdn-fp-inventory-plugin]$ grep module src/main/yang/l3vpn.yang

**module** l3vpn {

[tbk@vms-tme-centos2 myproject-tsdn-fp-inventory-plugin]$

How to find plan-data in yang module (find the path from highest level to a list where plan-data is found) :

[tbk@vms-tme-centos2 myproject-tsdn-fp-inventory-plugin]$ grep -B15 plan-data src/main/yang/l3vpn.yang

}

}

container vpn {

list l3vpn-plan {

config false;

tailf:cdb-oper {

tailf:persistent true;

}

key "name";

leaf name {

type string;

}

uses ncs:**plan-data**;

How to find service in yang module (find the path from highest level to a list where servicepoint is found) :

[tbk@vms-tme-centos2 myproject-tsdn-fp-inventory-plugin]$ grep -B30 servicepoint src/main/yang/l3vpn.yang

}

}

container vpn {

list l3vpn-plan {

config false;

tailf:cdb-oper {

tailf:persistent true;

}

key "name";

leaf name {

type string;

}

uses ncs:plan-data;

}

list l3vpn {

description "Layer3 VPN";

key name;

leaf name {

tailf:info "Unique service id";

tailf:cli-allow-range;

type string;

}

uses ncs:service-data;

ncs:**servicepoint** l3vpn-**servicepoint**;

### Edit the 2nd file MyprojectServiceInventoryPlugin.java

[tbk@vms-tme-centos2 myproject-tsdn-fp-inventory-plugin]$ grep myproject src/main/java/com/cisco/cw/tsdn/fp/myproject/inventory/MyprojectServiceInventoryPlugin.java

package com.cisco.cw.tsdn.fp.**myproject**.inventory;

\* **myproject** service inventory plugin implementation

private static final String SVC\_NS = "http://cisco.com/ns/cw/tsdn/service/**myproject**";

private static final String SVC\_MOD = "**myproject**";

private static final String SVC\_CONTAINER = "**myproject**";

private static final String SVC\_PLAN\_CONTAINER = "**myproject**-plan";

// **myproject**:**myproject**

// **myproject**:**myproject**-plan

return "**myproject**";

return "**myproject**";

[tbk@vms-tme-centos2 myproject-tsdn-fp-inventory-plugin]$

[tbk@vms-tme-centos2 myproject-tsdn-fp-inventory-plugin]$

See previous step to see how these values can be found in the yang file from NSO

SVC\_NS should be set to namespace

SVC\_MOD should be set to module

SVC\_CONTAINER should be set to the list where service is created

SVC\_PLAN\_CONTAINER should be set to the list where plan-data is created

Also find places where SVC\_CONTAINER and SVC\_PLAN\_CONTAINER are used and add vpn/ in the path. See below for the diffs.

16,23c16,23

< private static final String SVC\_NS = "http://cisco.com/ns/cw/tsdn/service/myproject";

< private static final String SVC\_MOD = "myproject";

< private static final String SVC\_CONTAINER = "myproject";

< private static final String SVC\_PLAN\_CONTAINER = "myproject-plan";

< // myproject:myproject

< private static final String SVC\_YANG\_PATH = SVC\_MOD + ":" + SVC\_CONTAINER;

< // myproject:myproject-plan

< private static final String SVC\_PLAN\_YANG\_PATH = SVC\_MOD + ":" + SVC\_PLAN\_CONTAINER;

---

> private static final String SVC\_NS = "http://com/example/l3vpn" ;

> private static final String SVC\_MOD = "l3vpn";

> private static final String SVC\_CONTAINER = "l3vpn";

> private static final String SVC\_PLAN\_CONTAINER = "l3vpn-plan";

> // tbk-test:tbk-test

> private static final String SVC\_YANG\_PATH = SVC\_MOD + ":vpn/" + SVC\_CONTAINER;

> // tbk-test:tbk-test-plan

> private static final String SVC\_PLAN\_YANG\_PATH = SVC\_MOD + ":vpn/" + SVC\_PLAN\_CONTAINER;

42c42

< return "myproject";

---

> return "SEVT-L3vpn";

47c47

< return "myproject";

---

> return "SEVT-L3vpn Description";

Note the following in the output below for the last few lines in the diff above:

ServiceTypeLabel is set to the label used on the service – in this case we are using SEVT-L3vpn.

ServiceTypeDescription is set to SEVT-L3vpn – any descriptive text may be provided here.

[tbk@vms-tme-centos2 myproject-tsdn-fp-inventory-plugin]$ grep -B3 -i myproject ./src/main/java/com/cisco/cw/tsdn/fp/myproject/inventory/MyprojectServiceInventoryPlugin.java

--

@Override

public String getServiceTypeLabel() {

return "myproject";

--

@Override

public String getServiceTypeDescription() {

return "myproject";

[tbk@vms-tme-centos2 myproject-t

### Final file versions

[tbk@vms-tme-centos2 myproject-tsdn-fp-inventory-plugin]$ cat ./src/main/java/com/cisco/cw/tsdn/fp/myproject/inventory/MyprojectServiceInventoryPlugin.java

/\*

\*/

package com.cisco.cw.tsdn.fp.myproject.inventory;

import javax.xml.namespace.QName;

import com.cisco.sp.cw.cat.nbi.restconf.common.api.AbstractServiceInventoryPlugin;

/\*\*

\* myproject service inventory plugin implementation

\*

\*/

public class MyprojectServiceInventoryPlugin extends AbstractServiceInventoryPlugin {

private static final String SVC\_NS = "http://com/example/l3vpn";

private static final String SVC\_MOD = "l3vpn";

private static final String SVC\_CONTAINER = "l3vpn";

private static final String SVC\_PLAN\_CONTAINER = "l3vpn-plan";

// myproject:myproject

private static final String SVC\_YANG\_PATH = SVC\_MOD + ":vpn/" + SVC\_CONTAINER;

// myproject:myproject-plan

private static final String SVC\_PLAN\_YANG\_PATH = SVC\_MOD + ":vpn/" + SVC\_PLAN\_CONTAINER;

@Override

public QName getServiceType() {

return new QName(SVC\_NS, SVC\_CONTAINER);

}

@Override

public String getServiceYangPath() {

return SVC\_YANG\_PATH;

}

@Override

public String getServicePlanYangPath() {

return SVC\_PLAN\_YANG\_PATH;

}

@Override

public String getServiceTypeLabel() {

return "SEVT-L3VPN";

}

@Override

public String getServiceTypeDescription() {

return "SEVT L3VPN Description";

}

@Override

public String getServiceYangModule() {

return SVC\_MOD;

}

@Override

public String getRootYangModule() {

// root yang module is same as service yang module as there is no augment in yangpath to service

// container

return SVC\_MOD;

}

}

[tbk@vms-tme-centos2 myproject-tsdn-fp-inventory-plugin]$ cat ./src/main/java/com/cisco/cw/tsdn/fp/myproject/inventory/MyprojectRestconfDataReader.java

/\*

\*/

package com.cisco.cw.tsdn.fp.myproject.inventory;

import java.sql.SQLException;

import java.util.Optional;

import org.opendaylight.mdsal.common.api.LogicalDatastoreType;

import org.opendaylight.yangtools.yang.data.api.YangInstanceIdentifier;

import org.opendaylight.yangtools.yang.data.api.schema.NormalizedNode;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import com.cisco.sp.cw.cat.nbi.restconf.common.api.Constants;

import com.cisco.sp.cw.cat.nbi.restconf.common.api.ServiceInventoryPlugin;

import com.cisco.sp.cw.cat.nbi.restconf.common.ctx.CatRestconfContext;

import com.cisco.sp.cw.cat.nbi.restconf.common.data.AbstractRestconfDataReader;

import com.cisco.sp.cw.cat.nbi.restconf.fw.store.PluggableDatastoreTransactionFactory;

import com.cisco.sp.cw.cat.nbi.restconf.fw.store.YangPath;

/\*\*

\* myproject restconf data reader implementation

\*

\*/

public class MyprojectRestconfDataReader extends AbstractRestconfDataReader {

private static final Logger LOG = LoggerFactory.getLogger(MyprojectRestconfDataReader.class);

public static PluggableDatastoreTransactionFactory<MyprojectRestconfDataReader> factory(

CatRestconfContext ctx,

ServiceInventoryPlugin serviceInventoryPlugin) {

return new PluggableDatastoreTransactionFactory<MyprojectRestconfDataReader>() {

@Override

public MyprojectRestconfDataReader newInstance() {

return new MyprojectRestconfDataReader(ctx, serviceInventoryPlugin);

}

@Override

public Class<MyprojectRestconfDataReader> getType() {

return MyprojectRestconfDataReader.class;

}

};

}

public MyprojectRestconfDataReader(CatRestconfContext ctx, ServiceInventoryPlugin serviceInventoryPlugin) {

super(ctx, serviceInventoryPlugin);

}

@YangPath(value = "(http://com/example/l3vpn?revision=2021-07-18)vpn/l3vpn", type = LogicalDatastoreType.OPERATIONAL)

public Optional<NormalizedNode<?, ?>> readMyprojectIntentData(YangInstanceIdentifier path) throws SQLException {

LOG.debug("Reading Service Intent data");

// if the key is other than name, change here.

return readServiceInstanceData(path, Constants.SERVICE\_YANG\_KEY);

}

@YangPath(value = "(http://com/example/l3vpn?revision=2021-07-18)vpn/l3vpn-plan", type = LogicalDatastoreType.OPERATIONAL)

public Optional<NormalizedNode<?, ?>> readMyprojectPlanData(YangInstanceIdentifier path) throws SQLException {

LOG.debug("Reading Service Plan data");

// if the key is other than name, change here.

return readServicePlanData(path, Constants.SERVICE\_YANG\_KEY);

}

}

[tbk@vms-tme-centos2 myproject-tsdn-fp-inventory-plugin]$

### Build inventory

[tbk@vms-tme-centos2 myproject-tsdn-fp-inventory-plugin]$ pwd

/home/tbk/cat-customization-doc/tsdn-fp-sdk/myproject-tsdn-fp/myproject-tsdn-fp-inventory-plugin

[tbk@vms-tme-centos2 myproject-tsdn-fp-inventory-plugin]$ build-prj

building tsdn tp maven project(s)...

[**INFO**] Scanning for projects...

Downloading from opendaylight-mirror: https://nexus.opendaylight.org/content/repositories/public/org/opendaylight/mdsal/binding-parent/6.0.7/binding-parent-6.0.7.pom

.. Takes 20 minutes or so to complete

[**INFO**] Installing com/cisco/cw/tsdn/fp/myproject/myproject-tsdn-fp-inventory-plugin/1.0.0-SNAPSHOT/myproject-tsdn-fp-inventory-plugin-1.0.0-SNAPSHOT.jar

[**INFO**] Writing OBR metadata

[**INFO**] **------------------------------------------------------------------------**

[**INFO**] **BUILD SUCCESS**

[**INFO**] **------------------------------------------------------------------------**

[**INFO**] Total time: 42:22 min

[**INFO**] Finished at: 2022-03-30T19:50:51-04:00

[**INFO**] **------------------------------------------------------------------------**

[tbk@vms-tme-centos2 myproject-tsdn-fp-inventory-plugin]$

## Edit myproject-tsdn-fp-package service-meta-data.json

### Find the file

[tbk@vms-tme-centos2 myproject-tsdn-fp]$ pwd

/home/tbk/cat-customization-doc/tsdn-fp-sdk/myproject-tsdn-fp

[tbk@vms-tme-centos2 myproject-tsdn-fp]$ ls -rlt

total 4

-rw-rw-r-- 1 tbk tbk 1204 Mar 30 18:03 pom.xml

drwxrwxr-x 3 tbk tbk 30 Mar 30 18:03 myproject-tsdn-fp-inventory-plugin

drwxrwxr-x 3 tbk tbk 30 Mar 30 18:03 myproject-tsdn-fp-overlay-plugin

drwxrwxr-x 3 tbk tbk 30 Mar 30 18:03 myproject-tsdn-fp-package

[tbk@vms-tme-centos2 myproject-tsdn-fp]$ cd myproject-tsdn-fp-package/

[tbk@vms-tme-centos2 myproject-tsdn-fp-package]$ find .

.

./pom.xml

./src

./src/assembly

./src/assembly/bin.xml

./src/main

./src/main/resources

./src/main/resources/tsdn-fp-descriptor.json

./src/main/resources/service-metadata.json

[tbk@vms-tme-centos2 myproject-tsdn-fp-package]$ cat src/main/resources/service-metadata.json

{

"services": {

"service-metadata": [

{

"model-version": "2021-07-18",

"namespace": "http://com/example/l3vpn" ,

"service-path": "l3vpn:vpn/l3vpn",

"plan-path": "l3vpn:vpn/l3vpn-plan",

"ui": {

"visible": "true"

},

"service-layer": "VPN",

"label": "SEVT Demo",

"service-type-label": "SEVT-L3vpn"

}

]

}

}

[tbk@vms-tme-centos2 myproject-tsdn-fp-package]$

### Make edits to match the NSO yang file

See previous steps for Restconf settings

Model-version is set same as revision in yang module

Namespace is set to the yang module namespace

Service yangpath and plan-paths from yang module

Ui visible to hide or expose

Service-layer VPN will show in the VPN services view on CNC UI. TRANSPORT will show in Traffic Engineering UI view.

Label shows on the provisioning UI the name of folder.

Service-type-label shows the service in provisioning UI under the folder

### Final diffs to review

5,15c5,8

< "model-version": "2021-07-18" ,

< "namespace": "http://com/example/l3vpn" ,

< "service-path": "l3vpn:vpn/l3vpn",

< "plan-path": "l3vpn:vpn/l3vpn-plan",

< "ui": {

< "visible": "true"

< },

< "service-layer": "VPN",

< "label": "SEVT Demo",

< "service-type-label": "SEVT-L3vpn"

<

---

> "model-version": "2022-03-30",

> "namespace": "http://cisco.com/ns/cw/tsdn/service/myproject",

> "service-path": "myproject:myproject",

> "plan-path": "myproject:myproject-plan"

### CNC UI Provisioning View

Graphical user interface, application, Teams

Description automatically generated

Graphical user interface, application

Description automatically generated

## Service Overlay Plugin Configuration

### Find files to edit

[tbk@vms-tme-centos2 myproject-tsdn-fp]$ pwd

/home/tbk/cat-customization-doc/tsdn-fp-sdk/myproject-tsdn-fp

[tbk@vms-tme-centos2 myproject-tsdn-fp]$ ls -rlt

total 4

-rw-rw-r-- 1 tbk tbk 1204 Mar 30 18:03 pom.xml

drwxrwxr-x 3 tbk tbk 30 Mar 30 18:03 myproject-tsdn-fp-inventory-plugin

drwxrwxr-x 3 tbk tbk 30 Mar 30 18:03 myproject-tsdn-fp-overlay-plugin

drwxrwxr-x 3 tbk tbk 30 Mar 30 18:03 myproject-tsdn-fp-package

[tbk@vms-tme-centos2 myproject-tsdn-fp]$ cd myproject-tsdn-fp-overlay-plugin/

[tbk@vms-tme-centos2 myproject-tsdn-fp-overlay-plugin]$ ls -rlt

total 8

-rw-rw-r-- 1 tbk tbk 4281 Mar 30 18:03 pom.xml

drwxrwxr-x 4 tbk tbk 28 Mar 30 18:03 src

[tbk@vms-tme-centos2 myproject-tsdn-fp-overlay-plugin]$ find . -name '\*.java'

./src/main/java/com/cisco/sp/cw/service/overlay/parsers/MyprojectParser.java

./src/test/java/com/cisco/cw/tsdn/fp/myproject/overlay/MyprojectTest.java

[tbk@vms-tme-centos2 myproject-tsdn-fp-overlay-plugin]$

### Service JSON from NSO

From NSO CLI:

admin@ncs# show running-config l3vpn:vpn l3vpn volvo | display json

{

"data": {

"l3vpn:vpn": {

"l3vpn": [

{

"name": "volvo",

"route-distinguisher": 999,

"endpoint": [

{

"id": "branch-office1",

"ce-device": "cpe-2.cisco.com",

"pe-device": "Node-4",

"ce-interface": "GigabitEthernet0/1.101",

"ip-network": "20.1.2.0/24",

"bandwidth": 600000,

"as-number": 65003

},

{

"id": "main-office",

"ce-device": "cpe-1",

"pe-device": "Node-5",

"ce-interface": "GigabitEthernet0/1.101",

"ip-network": "20.1.1.0/24",

"bandwidth": 1200000,

"as-number": 65003

}

]

}

]

}

}

}

Or by querying CNC via API (you can also query NSO restconf api directly):

curl -k --location --request GET 'https://198.18.134.219:30603/crosswork/proxy/nso/restconf/data/l3vpn:vpn/l3vpn=volvo?content=config' \

--header 'Content-Type: application/yang-data+json' \

--header 'Accept: application/yang-data+json' \

--header 'Authorization: Bearer <Token>'

{

"l3vpn:l3vpn": [

{

"name": "volvo",

"route-distinguisher": 999,

"endpoint": [

{

"id": "branch-office1",

"ce-device": "cpe-2.cisco.com",

"pe-device": "Node-4",

"ce-interface": "GigabitEthernet0/1.101",

"ip-network": "20.1.2.0/24",

"bandwidth": 600000,

"as-number": 65003

},

{

"id": "main-office",

"ce-device": "cpe-1",

"pe-device": "Node-5",

"ce-interface": "GigabitEthernet0/1.101",

"ip-network": "20.1.1.0/24",

"bandwidth": 1200000,

"as-number": 65003

}

]

}

]

}

### Overlay Model to populate

This is what the overlay plugin code needs to create from above.

{

   "name":"volvo",

   "yangPath":"l3vpn:l3vpn",

   "serviceType":"MP2MP",

   "virtualOverlayConfiguration":{

      "hosts":[

         "Node-4",

         "Node-5",

      ],

      "virtualDomains":[

         {

            "hostName":"Node-4",

            "caption":"ipv4",

            "decoration":"999:1",

            "edgeDirection":"BOTH"

         },

         {

            "hostName":"Node-5",

            "caption":"ipv4",

            "decoration":"999:1",

            "edgeDirection":"BOTH"

         }

      ]

   },

   "physicalOverlayConfiguration":{

      "srtePolicyPaths":[

         {

            "headEnd":"Node-4",

            "tailEnd":"Node-5",

            "connectionIdentifier":4110

         },

         {

            "headEnd":"Node-5",

            "tailEnd":"Node-4",

            "connectionIdentifier":4110

         }

      ],

      "rsvpteOverlayConfigurations":[

      ]

   },

   "errorInfo":[

   ]

}

Notes:

serviceType defines the type of visualization – P2P is point to point and MP is multipoint (for L3VPN and E-LAN etc)

virtualOverlayConfiguration/hosts : is a list of nodes participating in this VPN.

virtualOverlayConfiguration/virtualDomains: For each VPN router, shows what are being imported and exported:

caption: indicating ipv4 or ipv6

decoration: showing import export route-targets or whatever

edgeDirection: IN, OUT or BOTH – usually used for export, import of route-targets.

PhysicalOverlayConfigurations: Bind to sr-policies (headend,tailend,color) or rsvpte.

Headend, tailend are usually combinations of PE nodes (hosts in above terminology) and color is determined by "analyzing" the policy attached to VPNs which mark colors in the vrf export.

Graphical user interface

Description automatically generated

### How to find values in this example using jsonpath

# Use tools to find jsonPaths that give you the values

Example: <https://jsonpath.curiousconcept.com/>

Or

<https://jsonpath.herokuapp.com/>

Graphical user interface, text, application

Description automatically generated

Service name : $.\*[\*].name

# returns array of names, so use [0]

Hosts: $.\*[\*].endpoint[\*].pe-device

# returns array of pe-devices, which should be used as hosts

Caption: ipv4 (model needs to indicate ipv4 and ipv6 separately, in this example assume ip-network means ipv4)

Decoration: create from route-distinguisher which can be found by: $.\*[\*].route-distinguisher

#array of one value is returned in jsonpath query, so use [0]

Assume Both import/export is done with same, this is what this simple example model implements.

Note, see examples in the detailed wiki documents to see more complex examples, for instance how to find the preferred-path policy name from l2vpn and then find the color from sr-te policy etc. The CNC3.0 link is:

<https://wiki.cisco.com/display/ROBOTDEV/CNC+3.0+Service+Extensiblity+Developer+Guide>

### Edit MyprojectParser.java

There are number of edits done, the final file is shown.

You can also view the example file included to get more information – this is in examples/cat-fp folder in the $TSDN\_FP\_SDK\_HOME (See below cat command)

Note this is showing how to use Java to do the overlay plugin. Python based plugin is also supported, refer to documentation.

[tbk@vms-tme-centos2 parsers]$ pwd

/home/tbk/cat-customization-doc/tsdn-fp-sdk/myproject-tsdn-fp/myproject-tsdn-fp-overlay-plugin/src/main/java/com/cisco/sp/cw/service/overlay/parsers

[tbk@vms-tme-centos2 parsers]$

[tbk@vms-tme-centos2 parsers]$ cat $TSDN\_FP\_SDK\_HOME/examples/cat-fp/my-l3vpn-tsdn-fp/my-l3vpn-tsdn-fp-overlay-plugin/src/main/java/com/cisco/sp/cw/service/overlay/parsers/My\_l3vpnParser.java .

[tbk@vms-tme-centos2 parsers]$

[tbk@vms-tme-centos2 parsers]$ cp MyprojectParser.java MyprojectParser.java.bak

[tbk@vms-tme-centos2 parsers]$ ls -rlt

total 8

-rw-rw-r-- 1 tbk tbk 1542 Mar 30 19:43 MyprojectParser.java.bak

-rw-rw-r-- 1 tbk tbk 1542 Mar 30 19:43 MyprojectParser.java

[tbk@vms-tme-centos2 parsers]$ src/main/java/com/cisco/sp/cw/service/overlay/parsers

Make edits similar to the file below

[tbk@vms-tme-centos2 parsers]$ cat MyprojectParser.java

/\*

\*/

package com.cisco.sp.cw.service.overlay.parsers;

import java.util.Iterator;

import java.util.Map;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.stereotype.Component;

import com.cisco.cw.service.overlay.model.IServiceOverlay;

import com.cisco.cw.service.overlay.model.ServiceOverlay;

import com.cisco.cw.service.overlay.model.ServiceOverlayHelper;

import com.cisco.cw.service.overlay.model.ServiceType;

import com.cisco.cw.service.overlay.model.VirtualEdgeDirection;

import com.cisco.sp.cw.service.overlay.OverlayParser;

import com.cisco.cw.service.overlay.model.OverlayErrorInfo;

import com.cisco.cw.service.overlay.model.OverlayErrorType;

import com.cisco.cw.service.overlay.model.SRTEOverlayConfiguration;

import com.jayway.jsonpath.Configuration;

import com.jayway.jsonpath.DocumentContext;

import com.jayway.jsonpath.JsonPath;

import com.jayway.jsonpath.Option;

import lombok.extern.slf4j.Slf4j;

import net.minidev.json.JSONArray;

import net.minidev.json.JSONObject;

/\*\*

\* Demo l3vpn service overlay plugin implementation

\*

\*/

@Component

@Slf4j

public class MyprojectParser extends OverlayParser {

private static final Logger LOG = LoggerFactory.getLogger(MyprojectParser.class);

// EDIT Pattern for starting of json path

private static final String MY\_L3VPN\_JSON\_PATH = "$.\*[\*]";

// EDIT Pattern for getting hosts

private static final String ENDPOINTS\_JSON\_PATH = MY\_L3VPN\_JSON\_PATH + ".endpoint[\*]";

private Configuration configuration = Configuration.builder().options(Option.SUPPRESS\_EXCEPTIONS).build();

@Override

public String getType() {

// EDIT set name of module:service

return "l3vpn:vpn/l3vpn";

}

@Override

public IServiceOverlay parseOverlay(String serviceJson) {

// 1. parse service intent json to jsonpath context

DocumentContext jsonContext = JsonPath.using(configuration).parse(serviceJson);

// 2. build service virtual layers (key)

JSONArray nameArray = jsonContext.read(MY\_L3VPN\_JSON\_PATH + ".name");

String name = nameArray.get(0).toString() ;

String serviceKey = String.format("%s", name);

LOG.info("Building Service Overlay Model from service instance serviceKey={}", serviceKey);

ServiceOverlay serviceOverlay = ServiceOverlay.build(getType(), serviceKey);

serviceOverlay.setServiceType(ServiceType.MP);

/\* Sets the following values

"name":"volvo",

"yangPath":"l3vpn:vpn/l3vpn",

"serviceType":"MP2MP",

\*/

// 3. build service virtual layers (endpoints)

// In this example we are using route-targets as same value as rd:rd

JSONArray rdArray = jsonContext.read(MY\_L3VPN\_JSON\_PATH + ".route-distinguisher");

String rd = rdArray.get(0).toString() ;

// The NSO Model uses RT as RD:1

String routeTarget = String.format("%s:1",rd);

String caption = "ipv4" ;

JSONArray hosts = jsonContext.read(ENDPOINTS\_JSON\_PATH + ".pe-device");

for (Object hostObj : hosts) {

String host = (String)hostObj;

// Add for each host route-targets being imported, exported or both

// In this example we just have 1 route-target being both exported and imported for ipv4

serviceOverlay.addVirtualDomain((String)hostObj, caption, routeTarget, VirtualEdgeDirection.BOTH);

/\* Sets the hosts and virtualDomains

\*

"hosts":[

"Node-4",

"Node-5",

],

"virtualDomains":[

{

"hostName":"Node-4",

"caption":"ipv4",

"decoration":"999:1",

"edgeDirection":"BOTH"

},

{

"hostName":"Node-5",

"caption":"ipv4",

"decoration":"999:1",

"edgeDirection":"BOTH"

}

]

\*/

// for each pair of routers create underlay/transport paths

for (Object tailEndObj : hosts) {

String tailEnd = (String)tailEndObj;

// if headend and tailend are same no need to add

if ( host == tailEnd) continue ;

parseTransportLayer(serviceOverlay, jsonContext, host, tailEnd);

}

}

// 4. build service transport layer (sr policy headend, color, endpoint)

// parseTransportLayer(serviceOverlay, jsonContext, localNode, remoteNode);

LOG.info("##===>>> {}", serviceOverlay);

return serviceOverlay;

}

public static void parseTransportLayer(ServiceOverlay serviceOverlay, DocumentContext jsonContext, String host, String tailEnd) {

for (Object color : getColors(jsonContext)) {

SRTEOverlayConfiguration localSrtePolicy = new SRTEOverlayConfiguration(host,tailEnd,Long.valueOf(String.valueOf(color)));

serviceOverlay.addSRTEOverlayConfiguration(localSrtePolicy);

/\* This part is populated by above line

"physicalOverlayConfiguration":{

"srtePolicyPaths":[

{

"headEnd":"Node-4",

"tailEnd":"Node-5",

"connectionIdentifier":4110

},

{

"headEnd":"Node-5",

"tailEnd":"Node-4",

"connectionIdentifier":4110

}

],

}

\*/

}

}

# This example Hardcodes color to 4110. Code would need to be written

# to look at the export policy and figure out colors and then return those

# values for On demand next hop. For preferred-path, similarly code would need

# to be writted to get static policyname, then look at sr-te service with that

# name, and get color. See example code in the developer guide link.

public static JSONArray getColors (DocumentContext jsonContext) {

JSONArray colors=new JSONArray() ;

colors.add("4110");

return colors;

}

}

[tbk@vms-tme-centos2 parsers]$

## Build complete CNC CAT package

Run the build-prj from project folder, any errors during compilation need to be fixed.

[tbk@vms-tme-centos2 parsers]$ pwd

/home/tbk/cat-customization-doc/tsdn-fp-sdk/myproject-tsdn-fp/myproject-tsdn-fp-overlay-plugin/src/main/java/com/cisco/sp/cw/service/overlay/parsers

[tbk@vms-tme-centos2 parsers]$

[tbk@vms-tme-centos2 parsers]$ cd $TSDN\_FP\_SDK\_HOME/my\*

[tbk@vms-tme-centos2 myproject-tsdn-fp]$ pwd

/home/tbk/cat-customization-doc/tsdn-fp-sdk/myproject-tsdn-fp

[tbk@vms-tme-centos2 myproject-tsdn-fp]$

[tbk@vms-tme-centos2 myproject-tsdn-fp]$ build-prj

building tsdn tp maven project(s)...

[**INFO**] Scanning for projects...

..

[**INFO**] **Reactor Summary:**

[**INFO**]

[**INFO**] myproject-tsdn-fp-inventory-plugin ................. **SUCCESS** [ 16.794 s]

[**INFO**] myproject-tsdn-fp-overlay-plugin ................... **SUCCESS** [10:46 min]

[**INFO**] myproject-tsdn-fp-package .......................... **SUCCESS** [10:47 min]

[**INFO**] myproject-tsdn-fp-pom-aggregator 1.0.0-SNAPSHOT .... **SUCCESS** [ 0.006 s]

[**INFO**] **------------------------------------------------------------------------**

[**INFO**] **BUILD SUCCESS**

[**INFO**] **------------------------------------------------------------------------**

[**INFO**] Total time: 21:51 min

[**INFO**] Finished at: 2022-03-30T20:35:39-04:00

[**INFO**] **------------------------------------------------------------------------**

[tbk@vms-tme-centos2 myproject-tsdn-fp]$

[tbk@vms-tme-centos2 myproject-tsdn-fp-package]$ pwd

/home/tbk/cat-customization-doc/tsdn-fp-sdk/myproject-tsdn-fp/myproject-tsdn-fp-package

[tbk@vms-tme-centos2 myproject-tsdn-fp-package]$ ls -lrt

total 8

-rw-rw-r-- 1 tbk tbk 4583 Mar 30 18:03 pom.xml

drwxrwxr-x 4 tbk tbk 32 Mar 30 18:03 src

drwxrwxr-x 4 tbk tbk 120 Mar 30 20:35 target

[tbk@vms-tme-centos2 myproject-tsdn-fp-package]$ cd target/

[tbk@vms-tme-centos2 target]$

[tbk@vms-tme-centos2 target]$ ls -lrt

total 124

drwxrwxr-x 2 tbk tbk 102 Mar 30 20:35 dependency-maven-plugin-markers

drwxrwxr-x 2 tbk tbk 6 Mar 30 20:35 archive-tmp

-rw-rw-r-- 1 tbk tbk 126934 Mar 30 20:35 myproject-tsdn-fp-package-1.0.0-SNAPSHOT-bin.tar.gz

[tbk@vms-tme-centos2 target]$

## Upload and Deploy CNC CAT package

This is done using APIs. There is a two step process to get the bearer token that will be used.

First login to get ticket,

and then get jwt token.

Use the token for further API calls.

### Getting Ticket

# Replace username/password

curl --location --request POST 'https://198.18.134.219:30603/crosswork/sso/v1/tickets' \

--header 'Content-Type: application/x-www-form-urlencoded' \

--header 'Accept: text/plain' \

--data-raw 'username=admin&password=password'

Response is the ticket, use this in the next call to get token.

### Get Token

curl --location --request POST 'https://198.18.134.219:30603/crosswork/sso/v1/tickets/<ticket>' \

--header 'Content-Type: application/x-www-form-urlencoded' \

--data-raw 'service=https://198.18.134.219:30603/app-dashboard'

Response is the token

### Upload and deploy new CAT function pack.

Use token to upload file. Replace path of the file to appropriate path.

curl -k --location --request PATCH 'https://198.18.134.219:30603/crosswork/cat/cat-fp-deployment-manager-service/v1/twophasecommitrunner/uploadPackages' \

--header 'Authorization: Bearer <Token>’ \

--form 'packages=@"/Users/tbk/cat-customization/myproject-tsdn-fp-package-1.0.0-SNAPSHOT-bin.tar.gz"'

## Verify on CNC UI

Graphical user interface, application, map

Description automatically generated

Graphical user interface, application, map

Description automatically generated

Graphical user interface, application, map

Description automatically generated

Graphical user interface

Description automatically generated

Graphical user interface, application

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated