Genius : Oxygen Test Plan

Contents

Genius Integration and System Test

Code Name

The code name is odl-genius-ui. The integration test code is available under the genius folder of integration repo.

Description

Genius project provides Generic Network Interfaces, Utilities & Services. Any ODL application can use these to achieve interference-free co-existence with other applications using Genius.

In release cycle following enhancements and new features will be added -

1. Increase max services bind on an interface to 16
2. Service binding on tunnel interfaces
3. OF-based tunnels
4. TEP auto config

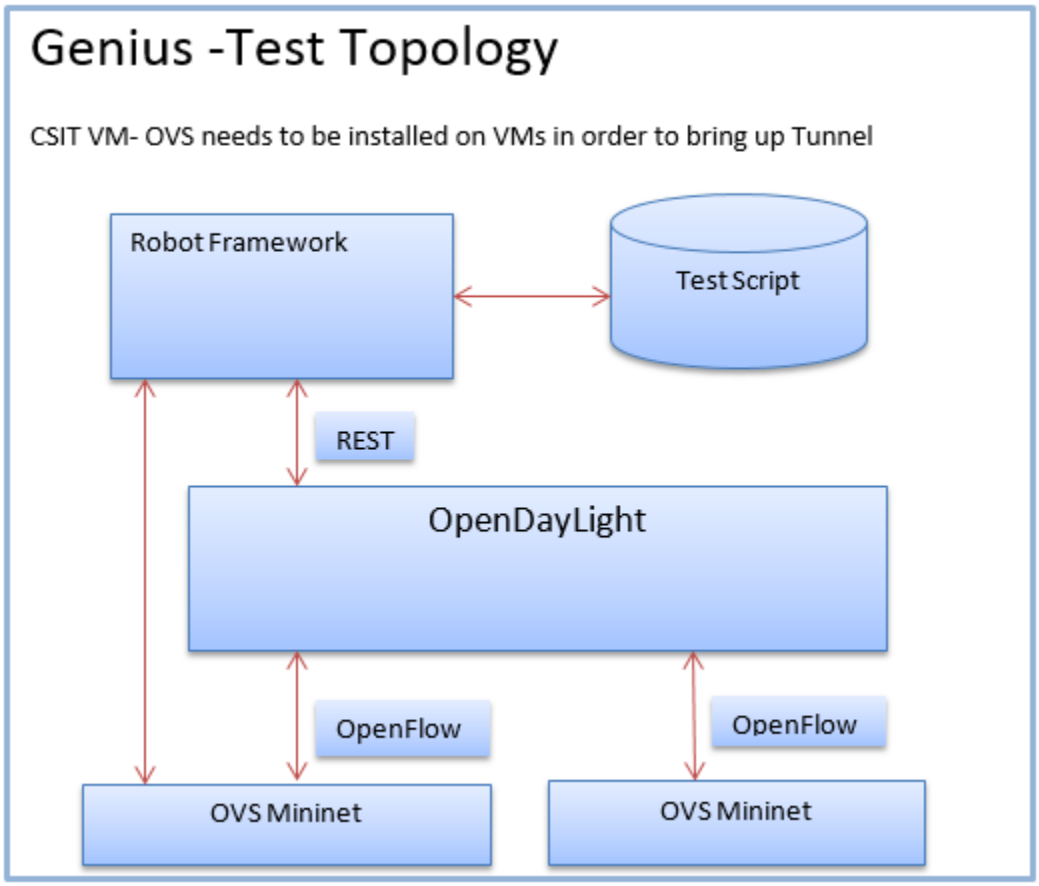
Apart from sanity and regression test of existing modules/features, additional tests will be added for verification of above mentioned features.

Overview & Design Architecture:

Please refer to design specifications of new features at following location -

http://docs.opendaylight.org/en/latest/submodules/genius/docs/index.html

Test Topology



Configuration

1. Install OVS 2.4 in the CSIT VM
2. Configuring Tunnels , Interfaces and Id manager is mentioned in respective testing scenarios

Bundles

- odl-genius-ui

Upstream Dependencies

Not Applicable

Downstream Dependencies

Not Applicable

Incompatibilities

Not Applicable

Network Intrusiveness

Not Applicable

Recommended Karaf features

Recommended features for karaf testing:

- odl-genius-ui

How to test

This feature provides REST to configure the ITM, Interface, ID manager. The tests have to include:

- Successful installation of the GENIUS feature and the dependencies.
- Create Bridges on both Ovs.
- Create tap port on one of the Ovs for testing Interface.
- Successful Creation of the ITM Tunnel between two OVS.
- Successful creation of Interfaces on OVS.
- Successful creation of ID pool as per test plan.
- Verification of data present in ODL data store after configuring.

Sample Test Configurations - REST

The Following example provides some of the REST Calls and their sample JSON data. The REST Calls can be made by the Openstack, if you have openstack installed and the ODL_MANAGER configured with the ODL IP, Otherwise you can achieve this from Postman or any similar REST Clients.

1. Create a Vxlan / GRE Tunnel: (Creates Internal Transport Manager between two OVS)

URL : http://localhost:8080/restconf/config/itm:transport-zones/

```
{
 "transport-zone": [
   {
     "zone-name": "transport zone name",
      "subnets": [
                {
                  "prefix":"subnet/24",
                  "vlan-id":0,
                  "vteps": [
                    {
                     "dpn-id": DPN 1 Id,
                     "portname": "port name",
                     "ip-address":"DPN 1 Ip address"
                    },
                    {
                     "dpn-id": DPN 2 Id,
                     "portname": "port name",
                     "ip-address":"DPN 2 Ip address"
                    }                   ],
     "gateway-ip":"gateway ip / 0.0.0.0"
     }
    ],
 "tunnel-type": "odl-interface:tunnel-type-vxlan/ odl-interface:tunnel-type-gre"
                    }
       ]
}
```

2. L2Vlan trunk Interface Creation :

URL: http://localhost:8080/restconf/config/ietf-interfaces:interfaces/

```
{
       "interface": [
           {
               "name": "interface name",
               "type": "iana-if-type:l2vlan",
               "l2vlan-mode":"trunk/transparent",
               "odl-interface:parent-interface": "tap port name",
               "enabled": "true"
          }]
}
```

Create member interface for the Parent interface created: (Creates a member interface for the trunk interface created using above Json)

URL : http://localhost:8080/restconf/config/ietf-interfaces:interfaces/

```
{
       "interface": [
           { "name": "member interface name",
               "type": "iana-if-type:l2vlan",
               "l2vlan-mode":"trunk-member",
               "vlan-id": Vlan Id,
               "odl-interface:parent-interface": "parent interface name",
               "enabled": "true"
             }
          ]
  }
```

3. Binding Service to the Parent interface created: (Binds service for the parent interface):

URL: http://localhost:8080/restconf/config/interface-service-bindings:service-bindings/services-info/{interface_name}/{service_mode}/

```
{
     "bound-services": [
       {
         "service-name": "service1",
         "flow-priority": "5",
         "service-type": "service-type-flow-based",
         "instruction": [
          {
           "order": 1,
           "go-to-table": {
              "table_id": 21
            }  }],
         "service-priority": "3",
         "flow-cookie": "1"
        } ,
        {
         "service-name": "service2",
         "flow-priority": "5",
         "service-type": "service-type-flow-based",
         "instruction": [
          {
           "order": 1,
           "go-to-table": {
              "table_id": 50
            }
         }],
         "service-priority": "4",
         "flow-cookie": "1"
       }
   ]
}
```

4. Unbind service: This URL unbinds the service which is binded

URL: http://localhost:8181/restconf/config/interface-service-bindings:service-bindings/services-info/{interface_name}/{service_mode}/bound-services/{service-priority}/

5. Creation of ID pool:

URL: http://localhost:8181/restconf/operations/id-manager:createIdPool

```
{
 "input": {
   "id-manager:pool-name": "test-pool",
   "id-manager:low": lower index,
   "id-manager:high":higher index
 }
}
```

6. Allocating ID Range:

URL: http://localhost:8181/restconf/operations/id-manager:allocateIdRange

```
{
 "input": {
   "id-manager:pool-name": "test-pool",
   "id-manager:id-key": "test-key", #different Id keys needs to be used if allocating different ranges
   "id-manager:size": size to be allocated
 }
}
```

7. Release Ids :

URL: http://localhost:8181/restconf/operations/id-manager:releaseId

```
{
 "input": {
   "id-manager:pool-name": "test-pool",
"id-manager:id-key": "test-key" # respective Id key needs to be used which is used in     allocateIdRange

  }
}
```

8. Delete ID pool:

URL: http://localhost:8181//restconf/operations/id-manager:deleteIdPool

```
{
 "input": {
   "id-manager:pool-name": "test-pool"
  }
}
```

Below are some of the REST interfaces, where you can find some useful information about the system.

1. GET IETF : http://localhost:8181/restconf/config/ietf-interfaces:interfaces/
2. GET IETF interface operational : http://localhost:8181/restconf/operational/ietf-interfaces:interfaces-state/
3. GET VPN interfaces config data store : http://localhost:8181/restconf/config/l3vpn:vpn-interfaces/
4. To Fetch DPN details : http://localhost:8181/restconf/operational/odl-interface-meta:bridge-ref-info/ </br>

Sample Test Configurations – OVS config CLIs

Please find below CLIs to Configure on OVS : Configure the below commands on the OVS using for testing. If 2 OVS is used, then both of them need to be configured using same commands below with changes needed as per the setup.

- sudo ovs-vsctl add-br bridge_name
- sudo ovs-vsctl set bridge bridge_name protocols=OpenFlow13
- sudo ovs-vsctl set-controller bridge_name tcp:${CONTROLLER_IP}:6633
- sudo ifconfig bridge_name up
- sudo ovs-vsctl add-port bridge_name tap_port -- set Interface tap_port type=tap
- sudo ovs-vsctl set-manager tcp:${CONTROLLER_IP}:6640

Performance/Scalability Concerns