Configuring Unmanaged Mode

About the Unmanaged Mode

The Layer 4 to Layer 7 service insertion feature enables an administrator to insert one or more services between two endpoint groups. The Application Policy Infrastructure Controller (APIC) allocates the fabric resources (VLANs) for the services and programs fabric (leaf switches) and service appliances as per the configuration specified in the service graph. The APIC requires the device package for a service to be able to use it as part of the service graph. The APIC also programs the service appliance during graph instantiation.

You might want the APIC to allocate only the network resources for the service graph and program only the fabric side during graph instantiation. This might be needed for various reasons, such as if your environment already has an existing orchestrator or a dev-op tool that is more suitable for programming the service appliance. In some other cases, the device package for the service appliance is not available.

The unmanaged mode for services enables you to choose the APIC's behavior for allocating network resources and programming the fabric. When enabled, the unmanaged mode restricts the APIC to allocate only the network resources for a service appliance and to program only the fabric (leaf). The configuration of the device is left to be done externally by you.

About Managed and Unmanaged Logical Devices

The unmanaged mode introduces the `managed` setting to the logical device (`LDevVip`), as shown in the following XML code:

```
<!-- Specified if the device is a managed device-->
<property name="managed"
```

```
        type="scalar:Bool"
        owner="management"
        mod="explicit">
        <default value="true"/>
</property>
```

A device can be either managed or unmanaged. When a device is configured as managed, the Application Policy Infrastructure Controller (APIC) manages the device and programs the device during graph instantiation. By default, when a device is registered with the Cisco APIC, the device is set to be in managed mode.

If a device is configured as unmanaged, meaning that the `managed` setting is set to `false`, the Cisco APIC does not program the device. The Cisco APIC only allocates the network resources and programs the VLAN/VXLAN on the fabric side.

The following settings are not needed when a device cluster is configured as unmanaged:

- Device package
- Connectivity information for the logical device (`vnsLDevViP`) and devices (`CDev`)—management IP address, credentials, and in-band connectivity information
- Information about supported function type (go-through, go-to, L1, L2)
- Information about context awareness (single context or multi-context)

The Cisco APIC still needs to know the topology information (`LIF`, `CIF`) for the logical device and devices. This information is needed so that the Cisco APIC can program the appropriate ports on the leaf and the Cisco APIC can also use this information for troubleshooting wizard purposes.

The Cisco APIC also needs to know the relation to `DomP`, which is used for allocating the encapsulation.

About Managed and Unmanaged Function Nodes

The unmanaged mode introduces the `managed` setting to the function node (`AbsNode`), as shown in the following XML code:

```
<!-- Specified if the function is using a managed device-->
<property name="managed"
              type="scalar:Bool"
              owner="management"
              mod="explicit">
              <default value="true"/>
</property>
```

A function node can be either managed or unmanaged. When a function node is configured as managed, the function node can use a managed device. The Application Policy Infrastructure Controller (APIC) programs the device during graph instantiation. By default, when a function node is added to the service graph, the function node is set to be in managed mode.

If a function node is configured as unmanaged, meaning that the `managed` setting is set to `false`, the APIC does not do parameter resolution nor program the devices. The APIC only allocates the network resources and programs the VLAN/VXLAN on fabric side.

The following settings are not needed when a function node is configured as unmanaged:

- `MFunc` relation
- `AbsFuncProfile`
- Configuration parameters (in `AbsNode` or on an endpoint group)

- Information about the supported function type (go-through, go-to)

The APIC still needs to know the network information (`LIF`, `CIF`) for the function node. This information is needed so that the APIC can program the network appropriately on the leaf, and the APIC can also use this information for troubleshooting wizard purposes.

The following settings are still needed:

- `LDevCtx` to enable the selection of `LDevVip` during graph instantiation
- `LIfCtx` to enable the selection of `LIf` during graph instantiation
- Bridge domain in `LIfCtx`
- Route peering in `LIfCtx`
- Subnet in `LIfCtx`

About Layer 4 to Layer 7 Services Endpoint Groups

As part of unmanaged mode feature, the Application Policy Infrastructure Controller (APIC) enables you to specify an endpoint group to be used for the graph connector during graph instantiation. This enables you to better troubleshoot the graph deployment. The APIC uses the Layer 4 to Layer 7 services endpoint group that you specified to download encapsulation information to a leaf. The APIC also uses the endpoint group to create port groups on the distributed virtual switch for virtual devices. A Layer 4 to Layer 7 services endpoint group is also used to aggregate faults and statistics information for a graph connector.

In addition to the enhanced visibility into the deployed graph resources, a Layer 4 to Layer 7 services endpoint group can also be used to specify static encapsulation to be used for a specific graph instance. This encapsulation can also be shared across multiple graph instances by sharing the Layer 4 to Layer 7 services endpoint group across multiple graph instances.

For example XML code that shows how you can use a Layer 4 to Layer 7 services endpoint group with a graph connector, see Example XML of Associating a Layer 4 to Layer 7 Service Endpoint Group with a Connector, on page 8.

Using Static Encapsulation for a Graph Connector

The Application Policy Infrastructure Controller (APIC) allocates the encapsulation for various service graphs during processing. In some use cases, you want to be able explicitly to specify the encapsulation to be used for a specific connector in the service graph. This is known as static encapsulation. Static encapsulations are only supported for the service graph connector that has a service device cluster with physical services. Service device clusters with virtual service devices use the dynamically allocated VLANs from the VMware domain that is associated with the service device cluster.

A static encapsulation can be used with a graph connector by specifying the encapsulation value as part of the Layer 4 to Layer 7 service endpoint group. For example XML code that shows how you can use a static encapsulation with a Layer 4 to Layer 7 services endpoint group, see Example XML of Using Static Encapsulation with a Layer 4 to Layer 7 Service Endpoint Group, on page 8.

Creating a Physical Device Using the NX-OS-Style CLI

This example procedure creates a physical device using the NX-OS-style CLI.

Step 1 Enter the configure mode.

Example:

```
apic1# configure
```

Step 2 Enter the configure mode for a tenant.

```
tenant tenant_name
```

Example:

```
apic1(config)# tenant t1
```

Step 3 Create a cluster:

Example:

```
apic1(config-tenant)# l4l7 cluster name ifav108-asa type physical vlan-domain phyDom5 servicetype FW
```

Step 4 Add a cluster device:

Example:

```
apic1(config-cluster)# cluster-device C1
```

Step 5 Add a provider cluster interface:

Example:

```
apic1(config-cluster)# cluster-interface provider
```

Step 6 Add a member device to the interface:

Example:

```
apic1(config-cluster-interface)# member device C1 device-interface Po1
apic1(config-member)# interface vpc VPCPolASA leaf 103 104
apic1(config-member)# exit
apic1(config-cluster-interface)# exit
```

Step 7 Add a consumer cluster interface:

Example:

```
apic1(config-cluster)# cluster-interface consumer
```

Step 8 Add the same member device to the consumer interface:

Example:

```
apic1(config-cluster-interface)# member device C1 device-interface Po1
apic1(config-member)# interface vpc VPCPolASA leaf 103 104
apic1(config-member)# exit
apic1(config-cluster-interface)# exit
```

Step 9 Exit out of the cluster creation mode:

Example:

```
apic1(config-cluster)# exit
```

Creating a High Availablity Cluster Using the NX-OS-Style CLI

This example procedure creates a high availability cluster using the NX-OS-style CLI.

Step 1 Enter the configure mode.

Example:

```
apic1# configure
```

Step 2 Enter the configure mode for a tenant.

```
tenant tenant_name
```

Example:

```
apic1(config)# tenant t1
```

Step 3 Create a cluster:

Example:

```
apic1(config-tenant)# l4l7 cluster name ifav108-asa type physical vlan-domain phyDom5 servicetype FW
```

Step 4 Add the cluster devices:

Example:

```
apic1(config-cluster)# cluster-device C1
apic1(config-cluster)# cluster-device C2
```

Step 5 Add a provider cluster interface:

Example:

```
apic1(config-cluster)# cluster-interface provider vlan 101
```

Step 6 Add member devices to the interface:

Example:

```
apic1(config-cluster-interface)# member device C1 device-interface Po1
apic1(config-member)# interface vpc VPCPolASA leaf 103 104
apic1(config-member)# exit
apic1(config-cluster-interface)# exit
apic1(config-cluster-interface)# member device C2 device-interface Po2
apic1(config-member)# interface vpc VPCPolASA-2 leaf 103 104
apic1(config-member)# exit
apic1(config-cluster-interface)# exit
```

Step 7 Add another provider cluster interface:

Example:

```
apic1(config-cluster)# cluster-interface provider vlan 102
```

Step 8 Add the same member devices from the first interface to this new interface:

Example:

```
apic1(config-cluster-interface)# member device C1 device-interface Po1
apic1(config-member)# interface vpc VPCPolASA leaf 103 104
apic1(config-member)# exit
apic1(config-cluster-interface)# exit
```

```
apic1(config-cluster-interface)# member device C2 device-interface Po2
apic1(config-member)# interface vpc VPCPolASA-2 leaf 103 104
apic1(config-member)# exit
apic1(config-cluster-interface)# exit
```

Step 9 Exit out of the cluster creation mode:

Example:

```
apic1(config-cluster)# exit
```

Creating a Virtual Device Using the NX-OS-Style CLI

This example procedure creates a virtual device using the NX-OS-style CLI.

Step 1 Enter the configure mode.

Example:

```
apic1# configure
```

Step 2 Enter the configure mode for a tenant.

```
tenant tenant_name
```

Example:

```
apic1(config)# tenant t1
```

Step 3 Create a cluster:

Example:

```
apic1(config-tenant)# l4l7 cluster name ifav108-citrix type virtual vlan-domain ACIVswitch servicetype
 ADC
```

Step 4 Add a cluster device:

Example:

```
apic1(config-cluster)# cluster-device D1 vcenter ifav108-vcenter vm NSVPX-ESX
```

Step 5 Add a consumer cluster interface:

Example:

```
apic1(config-cluster)# cluster-interface consumer
```

Step 6 Add a member device to the consumer interface:

Example:

```
apic1(config-cluster-interface)# member device D1 device-interface 1_1
apic1(config-member)# interface ethernet 1/45 leaf 102
ifav108-apic1(config-member)# vnic "Network adapter 2"
apic1(config-member)# exit
apic1(config-cluster-interface)# exit
```

Step 7 Add a provider cluster interface:

Example:

```
apic1(config-cluster)# cluster-interface provider
```

Step 8 Add the same member device to the provider interface:

Example:

```
apic1(config-cluster-interface)# member device D1 device-interface 1_1
apic1(config-member)# interface ethernet 1/45 leaf 102
ifav108-apic1(config-member)# vnic "Network adapter 2"
apic1(config-member)# exit
apic1(config-cluster-interface)# exit
```

Step 9 Exit out of the cluster creation mode:

Example:

```
apic1(config-cluster)# exit
```

Example XML for the Unmanaged Mode

The example XML in the following sections show how to administer the unmanaged mode.

Example XML of Creating an Unmanaged LDevVip Object

The following example XML creates an unmanaged `LDevVip` object:

```
<polUni>
    <fvTenant name="HA_Tenant1">
        <vnsLDevVip name="ADCCluster1" devtype="VIRTUAL" managed="no">
            <vnsRsALDevToDomP tDn="uni/vmmp-VMware/dom-mininet"/>
        </vnsLDevVip>
    </fvTenant>
</polUni>
```

For Cisco ACI Virtual Edge, the following example XML creates an unmanaged `LDevVip` object associated to the Cisco ACI Virtual Edge VMM domain with `ave` as the switching mode:

```
<polUni>
 <fvTenant name="HA_Tenant1">
  <vnsLDevVip name="ADCCluster1" devtype="VIRTUAL" managed="no">
   <vnsRsALDevToDomP switchingMode="AVE" tDn="uni/vmmp-VMware/dom-mininet_ave"/>
  </vnsLDevVip>
 </fvTenant>
</polUni>
```

Example XML of Creating an Unmanaged AbsNode Object

The following example XML creates an unmanaged `AbsNode` object:

```
    <fvTenant name="HA_Tenant1">
        <vnsAbsGraph name="g1">
            <vnsAbsTermNodeProv name="Input1">
                <vnsAbsTermConn name="C1">
                </vnsAbsTermConn>
            </vnsAbsTermNodeProv>

            <!-- Node1 provides a service function in un-managed mode -->
```

```
            <vnsAbsNode name="Node1" managed="no">
                <vnsAbsFuncConn name="outside" >
                </vnsAbsFuncConn>
                <vnsAbsFuncConn name="inside" >
                </vnsAbsFuncConn>
            </vnsAbsNode>

            <vnsAbsTermNodeCon name="Output1">
                <vnsAbsTermConn name="C6">
                </vnsAbsTermConn>
            </vnsAbsTermNodeCon>

            <vnsAbsConnection name="CON2" >
                <vnsRsAbsConnectionConns
tDn="uni/tn-HA_Tenant1/AbsGraph-g1/AbsTermNodeCon-Output1/AbsTConn"/>
                <vnsRsAbsConnectionConns
tDn="uni/tn-HA_Tenant1/AbsGraph-g1/AbsNode-Node1/AbsFConn-outside"/>
            </vnsAbsConnection>

            <vnsAbsConnection name="CON1" >
                <vnsRsAbsConnectionConns
tDn="uni/tn-HA_Tenant1/AbsGraph-g1/AbsNode-Node1/AbsFConn-inside"/>
                <vnsRsAbsConnectionConns
tDn="uni/tn-HA_Tenant1/AbsGraph-g1/AbsTermNodeProv-Input1/AbsTConn"/>
            </vnsAbsConnection>

        </vnsAbsGraph>
    </fvTenant>
```

Example XML of Associating a Layer 4 to Layer 7 Service Endpoint Group with a Connector

The following example XML associates a Layer 4 to Layer 7 service endpoint group with a connector:

```
<fvTenant name="HA_Tenant1">
   <vnsLDevCtx ctrctNameOrLbl="any" descr="" dn="uni/tn-HA_Tenant1/ldevCtx-c-any-g-any-n-any"

      graphNameOrLbl="any" name="" nodeNameOrLbl="any">
        <vnsRsLDevCtxToLDev tDn="uni/tn-HA_Tenant1/lDevVip-ADCCluster1"/>
        <vnsLIfCtx connNameOrLbl="inside" descr="" name="inside">
            <vnsRsLIfCtxToSvcEPg tDn="uni/tn-HA_Tenant1/ap-sap/SvcEPg-EPG1"/>
            <vnsRsLIfCtxToBD tDn="uni/tn-HA_Tenant1/BD-provBD1"/>
            <vnsRsLIfCtxToLIf tDn="uni/tn-HA_Tenant1/lDevVip-ADCCluster1/lIf-inside"/>
        </vnsLIfCtx>
        <vnsLIfCtx connNameOrLbl="outside" descr="" name="outside">
            <vnsRsLIfCtxToSvcEPg tDn="uni/tn-HA_Tenant1/ap-sap/SvcEPg-EPG2"/>
            <vnsRsLIfCtxToBD tDn="uni/tn-HA_Tenant1/BD-consBD1"/>
            <vnsRsLIfCtxToLIf tDn="uni/tn-HA_Tenant1/lDevVip-ADCCluster1/lIf-outside"/>
        </vnsLIfCtx>
    </vnsLDevCtx>
</fvTenant>
```

Example XML of Using Static Encapsulation with a Layer 4 to Layer 7 Service Endpoint Group

The following example XML uses static encapsulation with a Layer 4 to Layer 7 services endpoint group:

```
<polUni>
    <fvTenant name="HA_Tenant1">
```

```
        <fvAp name="sap">
            <vnsSvcEPg name="EPG1" encap="vlan-3510">
            </vnsSvcEPg>
        </fvAp>
    </fvTenant>
</polUni>
```

Unmanaged Mode Behavior

The following behavior applies regarding the unmanaged mode:

- Parameter resolution and unmanaged functions—For an unmanaged function, the Application Policy Infrastructure Controller (APIC) does not perform parameter resolution. Parameters are not required to be configured on `AbsGraph`, an endpoint group, or any other level.
- `VDev` and unmanaged functions— For an unmanaged function, the APIC does not perform parameter resolution or device side programming. No `VDev` tree is created for an unmanaged service graph function.
- Route peering in unmanaged mode—Unmanaged mode does not impact route peering functionality.
- VNIC auto-placement in unmanaged mode—Unmanaged mode does not impact VNIC placement functionality.