Layer 4 to Layer 7 Service Insertion

This chapter contains the following sections:

Layer 4 to Layer 7 Service Insertion

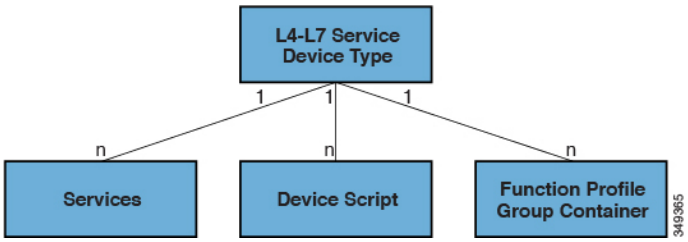
The Cisco Application Policy Infrastructure Controller (APIC) manages network services. Policies are used to insert services. APIC service integration provides a life cycle automation framework that enables the system to dynamically respond when a service comes online or goes offline. Shared services that are available to the entire fabric are administered by the fabric administrator. Services that are for a single tenant are administered by the tenant administrator.

The APIC provides automated service insertion while acting as a central point of policy control. APIC policies manage both the network fabric and services appliances. The APIC can configure the network automatically so that traffic flows through the services. Also, the APIC can automatically configure the service according to the application's requirements. This approach allows organizations to automate service insertion and eliminate the challenge of managing all of the complex traffic-steering techniques of traditional service insertion.

Layer 4 to Layer 7 Policy Model

The Layer 4 to Layer 7 service device type policies includes key managed objects such as services supported by the package and device scripts. The following figure shows the objects of the Layer 4 to Layer 7 service device type policy model.

Figure 1: Layer 4 to Layer 7 Policy Model



Layer 4 to Layer 7 service policies contain the following:

- **Services**—Contains metadata for all the functions provided by a device such as SSL offloading and load-balancing. This MO contains the connector names, encapsulation type, such as VLAN and VXLAN, and any interface labels.
- **Device Script**—Represents a device script handler that contains meta information about the related attributes of the script handler including its name, package name, and version.
- **Function Profile Group Container**—Objects that contain the functions available to the service device type. Function profiles contain all the configurable parameters supported by the device organized into folders.

About Service Graphs

The Cisco Application Centric Infrastructure (ACI) treats services as an integral part of an application. Any services that are required are treated as a service graph that is instantiated on the ACI fabric from the Cisco Application Policy Infrastructure Controller (APIC). Users define the service for the application, while service graphs identify the set of network or service functions that are needed by the application.

A service graph represents the network using the following elements:

- Function node—A function node represents a function that is applied to the traffic, such as a transform (SSL termination, VPN gateway), filter (firewalls), or terminal (intrusion detection systems). A function within the service graph might require one or more parameters and have one or more connectors.
- Terminal node—A terminal node enables input and output from the service graph.
- Connector—A connector enables input and output from a node.
- Connection—A connection determines how traffic is forwarded through the network.

After the graph is configured in the APIC, the APIC automatically configures the services according to the service function requirements that are specified in the service graph. The APIC also automatically configures the network according to the needs of the service function that is specified in the service graph, which does not require any change in the service device.

A service graph is represented as two or more tiers of an application with the appropriate service function inserted between.

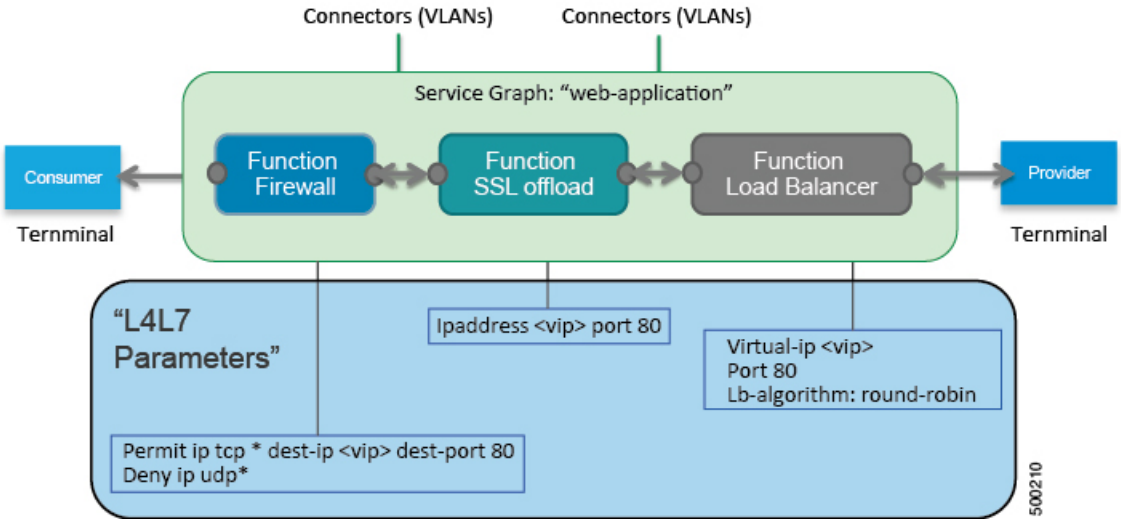
A service appliance (device) performs a service function within the graph. One or more service appliances might be required to render the services required by a graph. One or more service functions can be performed by a single-service device.

Service graphs and service functions have the following characteristics:

- Traffic sent or received by an endpoint group can be filtered based on a policy, and a subset of the traffic can be redirected to different edges in the graph.
- Service graph edges are directional.
- Taps (hardware-based packet copy service) can be attached to different points in the service graph.
- Logical functions can be rendered on the appropriate (physical or virtual) device, based on the policy.
- The service graph supports splits and joins of edges, and it does not restrict the administrator to linear service chains.
- Traffic can be reclassified again in the network after a service appliance emits it.
- Logical service functions can be scaled up or down or can be deployed in a cluster mode or 1:1 active-standby high-availability mode, depending on the requirements.

The following figure provides an example of a service graph deployment:

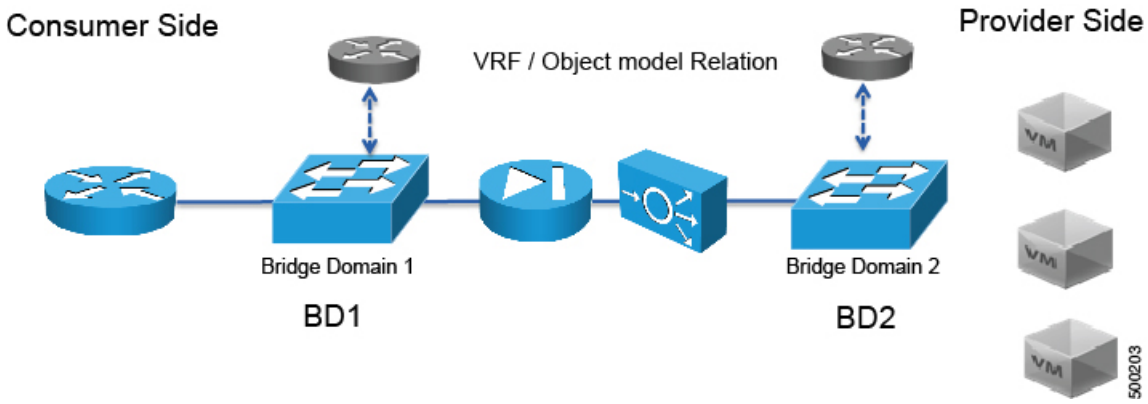
Figure 2: Example Service Graph Deployment



By using a service graph, you can install a service, such as an ASA firewall, once and deploy it multiple times in different logical topologies. Each time the graph is deployed, ACI takes care of changing the configuration on the firewall to enable the forwarding in the new logical topology.

Deploying a service graph requires bridge domains and VRFs, as shown in the following figure:

Figure 3: Bridge Domains and VRFs of a Service Graph



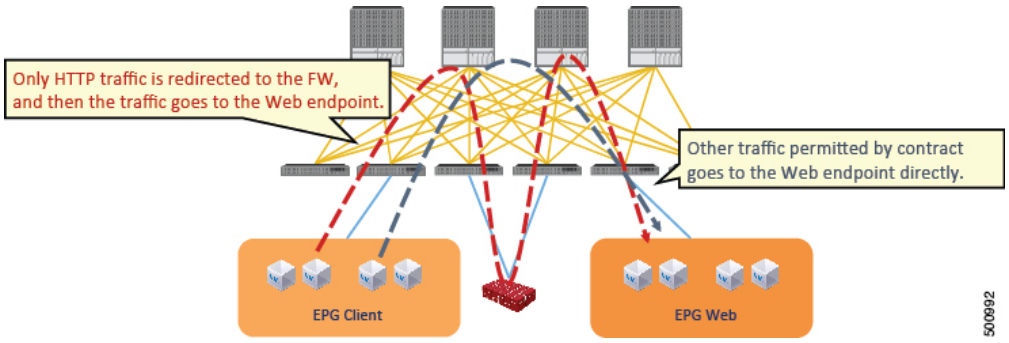
Note If you have some of the legs of a service graph that are attached to endpoint groups in other tenants, when you use the **Remove Related Objects of Graph Template** function in the GUI, the APIC does not remove contracts that were imported from tenants other than where the service graph is located. The APIC also does not clean endpoint group contracts that are located in a different tenant than the service graph. You must manually remove these objects that are in different tenants.

About Policy-Based Redirect

Cisco Application Centric Infrastructure (ACI) policy-based redirect (PBR) enables provisioning service appliances, such as firewalls or load balancers, as managed or unmanaged nodes without needing a Layer 4 to Layer 7 package. Typical use cases include provisioning service appliances that can be pooled, tailored to application profiles, scaled easily, and have reduced exposure to service outages. PBR simplifies the deployment of service appliances by enabling the provisioning consumer and provider endpoint groups to be all in the same virtual routing and forwarding (VRF) instance. PBR deployment consists of configuring a route redirect policy and a cluster redirect policy, and creating a service graph template that uses the route and cluster redirect policies. After the service graph template is deployed, use the service appliance by enabling endpoint groups to consume the service graph provider endpoint group. This can be further simplified and automated by using vzAny. While performance requirements may dictate provisioning dedicated service appliances, virtual service appliances can also be deployed easily using PBR.

The following figure illustrates the use case of redirecting specific traffic to the firewall:

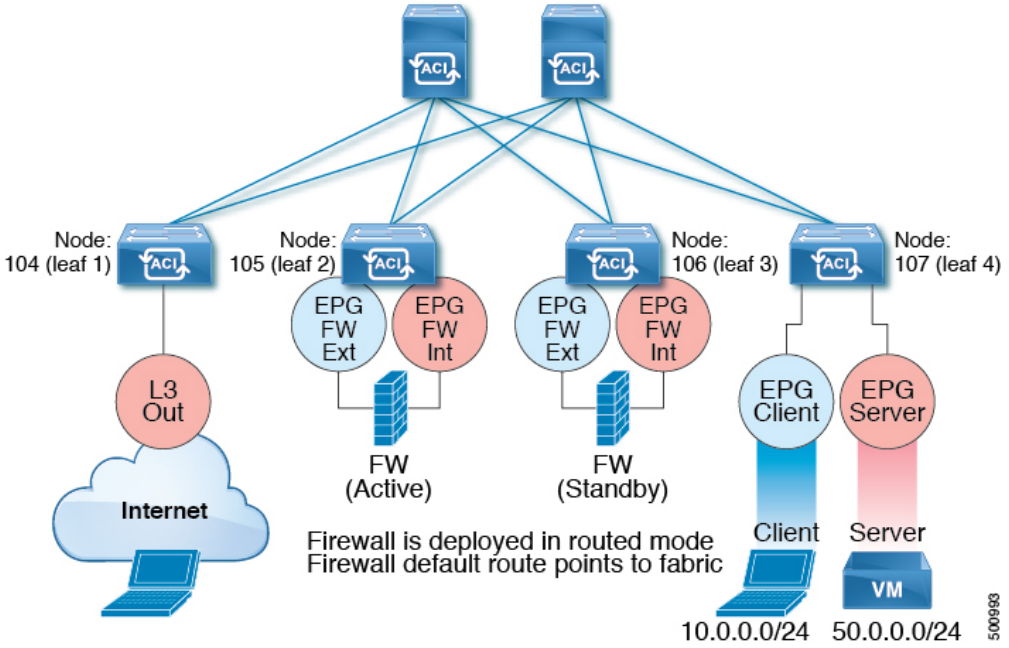
Figure 4: Use Case: Redirecting Specific Traffic to the Firewall



In this use case, you must create two subjects. The first subject permits HTTP traffic, which then gets redirected to the firewall. After the traffic passes through the firewall, it goes to the Web endpoint. The second subject permits all traffic, which captures traffic that is not redirected by the first subject. This traffic goes directly to the Web endpoint.

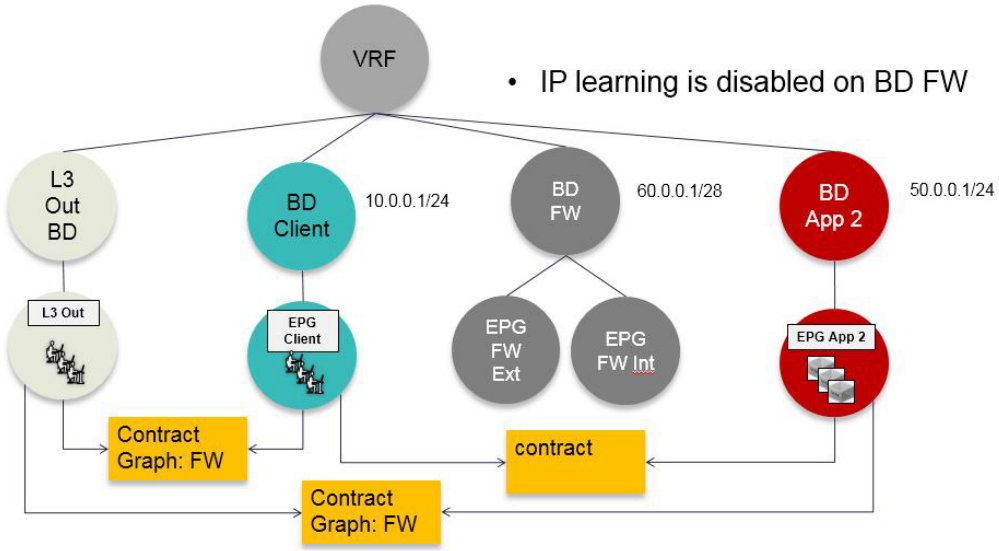
The following figure illustrates a sample ACI PBR physical topology:

Figure 5: Sample ACI PBR Physical Topology



The following figure illustrates a sample ACI PBR logical topology:

Figure 6: Sample ACI PBR Logical Topology



While these examples illustrate simple deployments, ACI PBR enables scaling up mixtures of both physical and virtual service appliances for multiple services, such as firewalls and server load balancers.

About Symmetric Policy-Based Redirect

Symmetric policy-based redirect (PBR) configurations enable provisioning a pool of service appliances so that the consumer and provider endpoint groups traffic is policy-based. The traffic is redirected to one of the service nodes in the pool, depending on the source and destination IP equal-cost multi-path routing (ECMP) prefix hashing.

Note Symmetric PBR configurations require 9300-EX hardware.

Sample symmetric PBR REST posts are listed below:

```
Under fvTenant svcCont

<vnsSvcRedirectPol name="LoadBalancer_pool">
          <vnsRedirectDest name="lb1" ip="1.1.1.1" mac="00:00:11:22:33:44"/>
          <vnsRedirectDest name="lb2" ip="2.2.2.2" mac="00:de:ad:be:ef:01"/>
          <vnsRedirectDest name="lb3" ip="3.3.3.3" mac="00:de:ad:be:ef:02"/>
</vnsSvcRedirectPol>

<vnsLIfCtx name="external">
          <vnsRsSvcRedirectPol tnVnsSvcRedirectPolName="LoadBalancer_pool"/>
          <vnsRsLIfCtxToBD tDn="uni/tn-solar/bd-fwBD">
</vnsLIfCtx>

<vnsAbsNode name="FW" routingMode="redirect">
```

Sample symmetric PBR NX-OS-style CLI commands are listed below.

The following commands under the tenant scope create a service redirect policy:

```
apic1(config-tenant)# svcredir-pol fw-external
apic1(svcredir-pol)# redir-dest 2.2.2.2 00:11:22:33:44:56
```

The following commands enable PBR:

```
apic1(config-tenant)# l4l7 graph FWOnly contract default
apic1(config-graph)# service FW svcredir enable
```

The following commands set the redirect policy under the device selection policy connector:

```
apic1(config-service)# connector external
apic1(config-connector)# svcredir-pol tenant solar name fw-external
```

Automated Service Insertion

Although VLAN and virtual routing and forwarding (VRF) stitching is supported by traditional service insertion models, the Application Policy Infrastructure Controller (APIC) can automate service insertion and the provisioning of network services, such as Secure Sockets Layer (SSL) offload, server load balancing (SLB), Web Application firewall (WAF), and firewall, while acting as a central point of policy control. The network services are typically rendered by service appliances, such as Application Delivery Controllers (ADCs) and firewalls. The APIC policies manage both the network fabric and services appliances. The APIC can configure the network automatically so that traffic flows through the services. The APIC can also automatically configure the service according to the application's requirements, which allows organizations to automate service insertion and eliminate the challenge of managing the complex techniques of traditional service insertion.

About Device Packages

The Application Policy Infrastructure Controller (APIC) requires a device package to configure and monitor service devices. You add service functions to the APIC through the device package. A device package manages a single class of service devices and provides the APIC with information about the device and its capabilities. A device package is a `zip` file that contains the following parts:

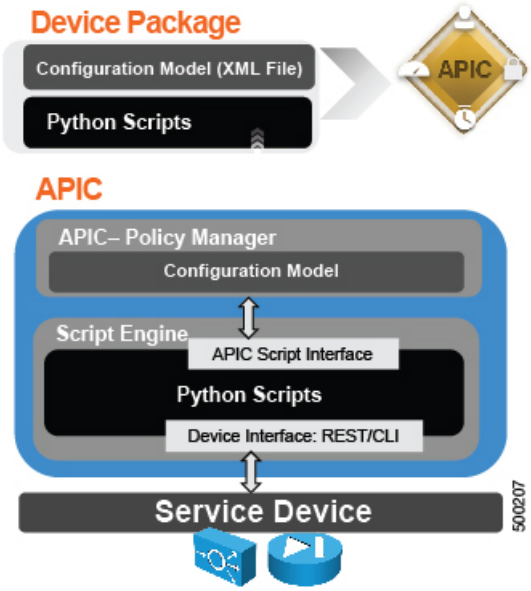
Device specification	An XML file that defines the following:
	• Device properties:
	  • **Model**—Model of the device.
	  • **Vendor**—Vendor of the device.
	  • **Version**—Software version of the device.
	• Functions provided by a device, such as load balancing, content switching, and SSL termination.
	• Interfaces and network connectivity information for each function.
	• Device configuration parameters.
	• Configuration parameters for each function.

Device script	A Python script that interacts with the device from the APIC. APIC events are mapped to function calls that are defined in the device script. A device package can contain multiple device scripts. A device script can interface with the device by using REST, SSH, or any similar mechanism.
Function profile	Function parameters with default values that are specified by the vendor. You can configure a function to use these default values.
Device-level configuration parameters	A configuration file that specifies parameters that are required by a device. This configuration can be shared by one or more graphs using a device.

You can create a device package or it can be provided by a device vendor or Cisco.

The following figure illustrates the interaction of a device package and the APIC:

Figure 7: Device Package and the APIC

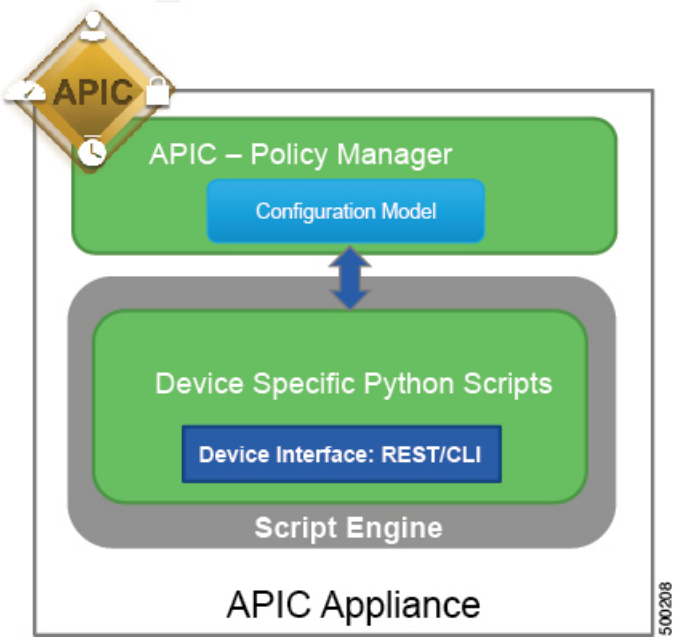


The functions in a device script are classified into the following categories:

- Device/Infrastructure—For device level configuration and monitoring
- Service Events—For configuring functions, such as a server load balancer or Secure Sockets Layer, on the device
- Endpoint/Network Events—For handling endpoint and network attach/detach events

The APIC uses the device configuration model that is provided in the device package to pass the appropriate configuration to the device scripts. The device script handlers interface with the device using its REST or CLI interface.

Figure 8: How the Device Scripts Interface with a Service Device



The device package enables an administrator to automate the management of the following services:

- Device attachment and detachment
- Endpoint attachment and detachment
- Service graph rendering
- Health monitoring
- Alarms, notifications, and logging
- Counters

For more information about device packages and how to develop a device package, see *Cisco APIC Layer 4 to Layer 7 Device Package Development Guide*

About Device Clusters

A device cluster (also known as a logical device) is one or more concrete devices that act as a single device. A device cluster has cluster (logical) interfaces, which describe the interface information for the device cluster. During service graph template rendering, function node connectors are associated with cluster (logical) interfaces. The Application Policy Infrastructure Controller (APIC) allocates the network resources (VLAN or Virtual Extensible Local Area Network [VXLAN]) for a function node connector during service graph template instantiation and rendering and programs the network resources onto the cluster (logical) interfaces.

The service graph template uses a specific device that is based on a device selection policy (called a *logical device context*) that an administrator defines.

An administrator can set up a maximum of two concrete devices in active-standby mode.

To set up a device cluster, you must perform the following tasks:

1. Connect the concrete devices to the fabric.
2. Assign the management IP address to the device cluster.
3. Register the device cluster with the APIC. The APIC validates the device using the device specifications from the device package.

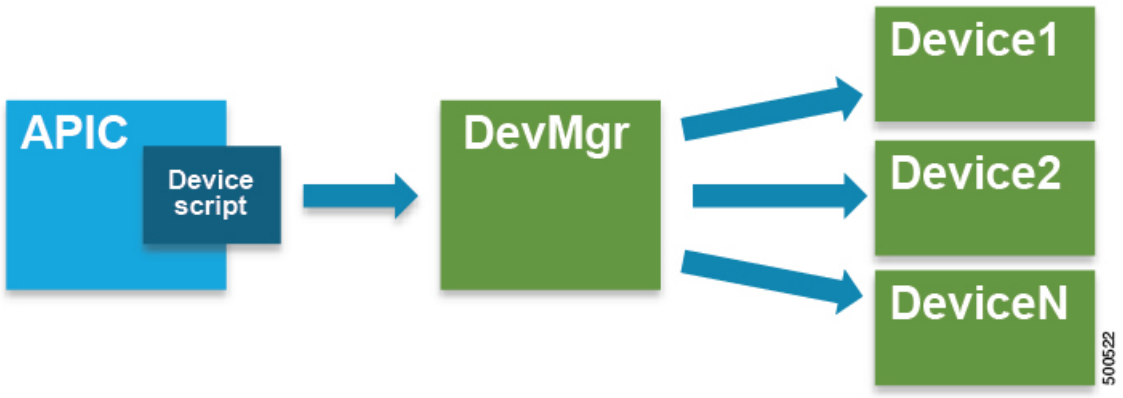
Note The APIC does not validate a duplicate IP address that is assigned to two device clusters. The APIC can provision the wrong device cluster when two device clusters have the same management IP address. If you have duplicate IP addresses for your device clusters, delete the IP address configuration on one of the devices and ensure there are no duplicate IP addresses that are provisioned for the management IP address configuration.

About Device Managers and Chassis Managers

A device manager serves as a single point of configuration for a set of clusters in a Cisco Application Centric Infrastructure (ACI) fabric. The administration or operational state is presented in the native GUI of the devices. A device manager handles configuration on individual devices and enables you to simplify the configuration in the Application Policy Infrastructure Controller (APIC). You create a template in device manager, then populate the device manager with instance-specific values from the APIC, which requires only a few values.

The following figure illustrates a device manager controlling multiple devices in a cluster:

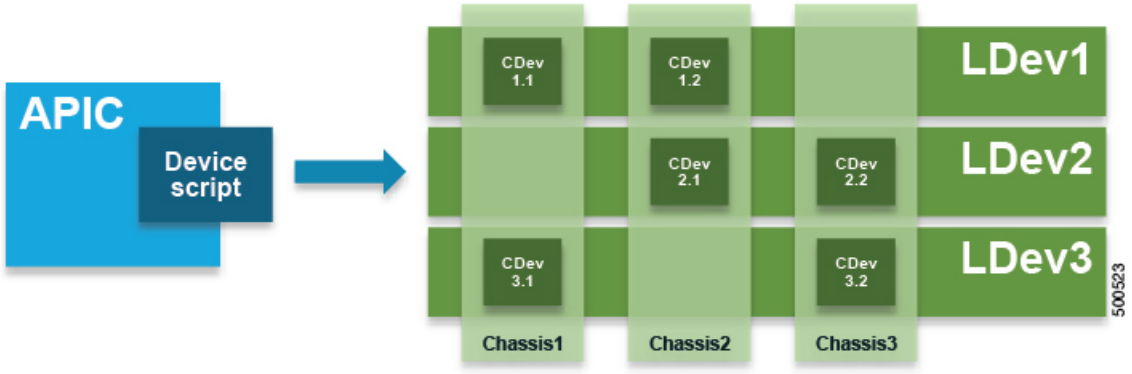
Figure 9: Controlling Devices with a Device Manager



A chassis manager is a physical or virtual "container" of processing resources. A chassis manager supports a number of virtual service devices that are represented as `CDev` objects. A chassis manager handles networking, while `CDev` handles processing. A chassis manager enables the on-demand creation of virtual processing nodes. For a virtual device, some parts of a service (specifically the VLANs) must be applied to the chassis rather than to the virtual machine. To accomplish this, the chassis management IP address and credentials must be included in callouts.

The following figure illustrates a chassis manager acting as a container of processing resources:

Figure 10: Controlling Devices with a Device Manager

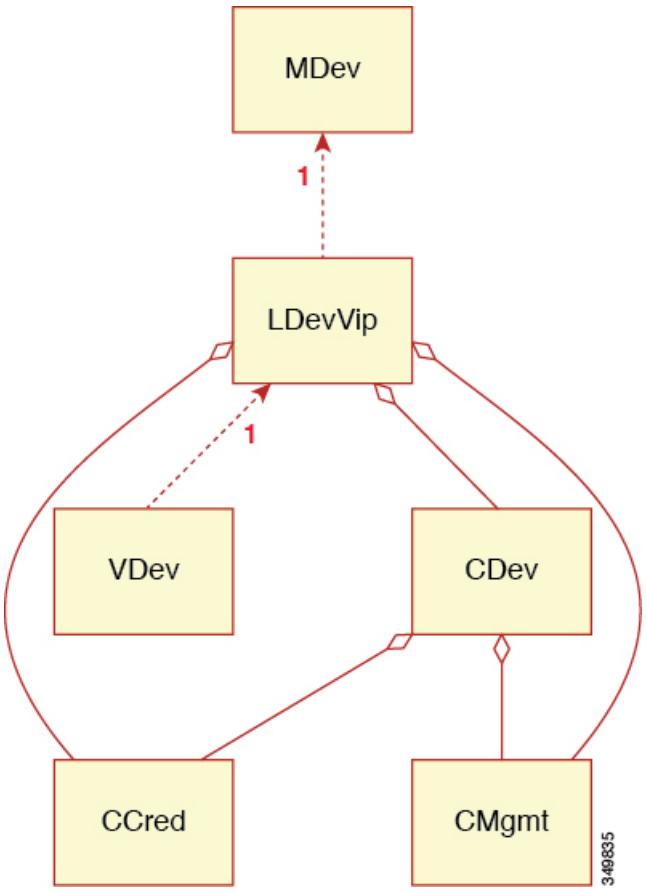


Without a device manager or chassis manager, the model for service devices contains the following key managed objects:

- `MDev`—Represents a device type (vendor, model, version).
- `LDevVIP`—Represents a cluster, a set of identically configured devices for Cold Standby. Contains `CMgmt` and `CCred` for access to the device.
- `CDev`—Represents a member of a cluster, either physical or virtual. Contains `CMgmt` and `CCred` for access to the device.
- `VDev`—Represents a context on a cluster, similar to a virtual machine on a server.

The following figure illustrates the model for the key managed objects, with `CMgmt` (management connectivity) and `CCred` (credentials) included:

Figure 11: Managed Object Model Without a Device Manager or Chassis Manager



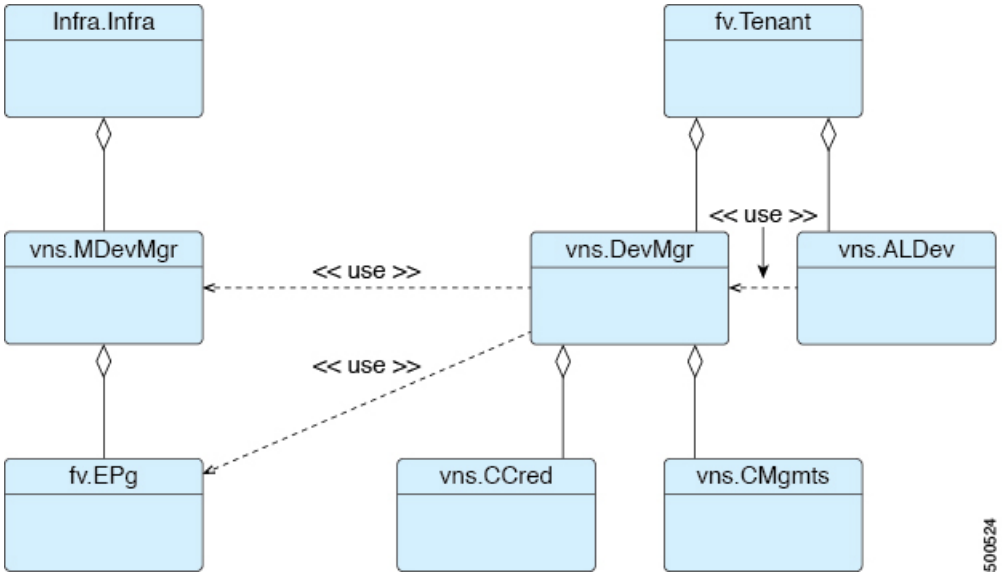
`CMgmt` (host + port) and `CCred` (username + password) allow the script to access the device and cluster.

A device manager and chassis manager adds the ability to control the configuration of clusters and devices from a central management station. The chassis adds a parallel hierarchy to the `MDev` object and `ALDev` object to allow a `CDev` object to be tagged as belonging to a specific chassis. The following managed objects are added to the model to support the device and chassis manager concept:

- `MDevMgr`—Represents a type of device manager. An `MDevMgr` can manage a set of different `MDev`s, which are typically different products from the same vendor.
- `DevMgr`—Represents a device manager. Access to the manager is provided using the contained `CMgmt` and `CCred` managed objects. Each cluster can be associated with only one `DevMgr`.
- `MChassis`—Represents a type of chassis. This managed object is typically included in the package.
- `Chassis`—Represents a chassis instance. It contains the `CMgmt` and `CCred`[Secret] managed objects to provide connectivity to the chassis.

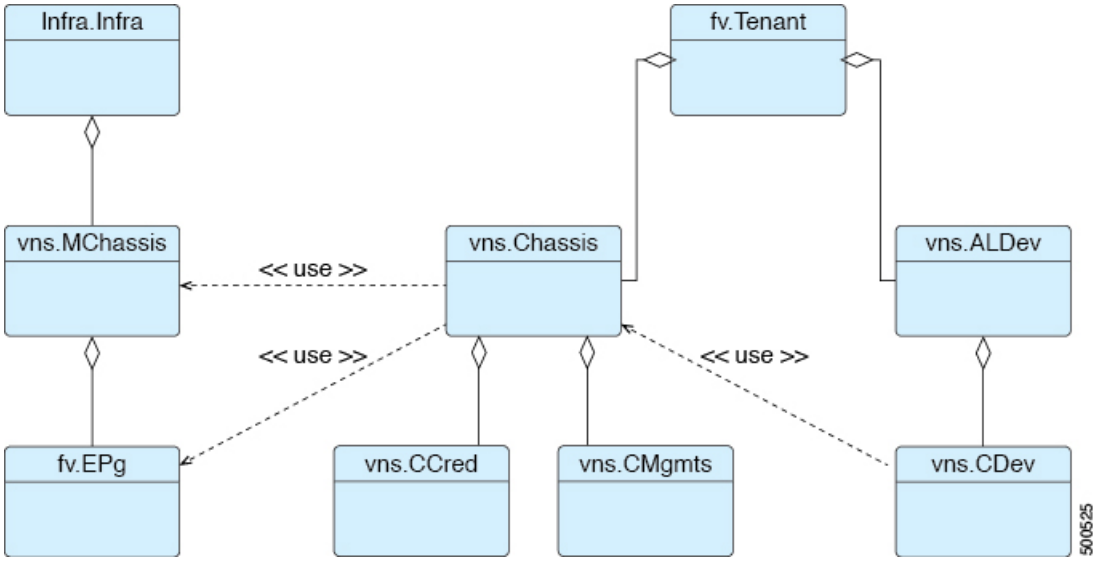
The following figure illustrates the device manager object model:

Figure 12: Device Manager Object Model



The following figure illustrates the chassis manager object model:

Figure 13: Chassis Manager Object Model



About Concrete Devices

A concrete device can be either physical or virtual. If the device is virtual, you must choose the controller (vCenter or SCVMM controller) and the virtual machine name. A concrete device has concrete interfaces. When a concrete device is added to a logical device, the concrete interfaces are mapped to the logical interfaces. During service graph template instantiation, VLANs and VXLANs are programmed on concrete interfaces that are based on their association with logical interfaces.

About Function Nodes

A function node represents a single service function. A function node has function node connectors, which represent the network requirement of a service function.

A function node within a service graph can require one or more parameters. The parameters can be specified by an endpoint group (EPG), an application profile, or a tenant VRF. Parameters can also be assigned at the time that you define a service graph. The parameter values can be locked to prevent any additional changes.

Note For Multi-Site configuration, up to 2 nodes can be deployed in a service graph. For non-Multi-Site configuration, up to 5 nodes can be deployed in a service graph.

About Function Node Connectors

A function node connector connects a function node to the service graph and is associated with the appropriate bridge domain and connections based on the graph's connector's subset. Each connector is associated with a VLAN or Virtual Extensible LAN (VXLAN). Each side of a connector is treated as an endpoint group (EPG), and whitelists are downloaded to the switch to enable communication between the two function nodes.

About Terminal Nodes

Terminal nodes connect a service graph with the contracts. You can insert a service graph for the traffic between two application endpoint groups (EPGs) by connecting the terminal node to a contract. Once connected, traffic between the consumer EPG and provider EPG of the contract is redirected to the service graph.

About Privileges

An administrator can grant privileges to the roles in the APIC. Privileges determine what tasks a role is allowed to perform. Administrators can grant the following privileges to the administrator roles:

Privilege	Description
nw-svc-connectivity	• Create a management EPG • Create management connectivity to other objects
nw-svc-policy	• Create a service graph • Attach a service graph to an application EPG and a contract • Monitor a service graph
nw-svc-device	• Create a device cluster • Create a concrete device • Create a device context

Note Only an infrastructure administrator can upload a device package to the APIC.

Service Automation and Configuration Management

The Cisco APIC can optionally act as a point of configuration management and automation for service devices and coordinate the service devices with the network automation. The Cisco APIC interfaces with a service device by using Python scripts and calls device-specific Python script functions on various events.

The device scripts and a device specification that defines functions supported by the service device are bundled as a device package and installed on the Cisco APIC. The device script handlers interface with the device by using its REST interface (preferred) or CLI based on the device configuration model.

Service Resource Pooling

The Cisco ACI fabric can perform nonstateful load distribution across many destinations. This capability allows organizations to group physical and virtual service devices into service resource pools, which can be further grouped by function or location. These pools can offer high availability by using standard high-availability mechanisms or they can be used as simple stateful service engines with the load redistributed to the other members if a failure occurs. Either option provides horizontal scale out that far exceeds the current limitations of the equal-cost multipath (ECMP), port channel features, and service appliance clustering, which requires a shared state.

Cisco ACI can perform a simple version of resource pooling with any service devices if the service devices do not have to interact with the fabric, and it can perform more advanced pooling that involves coordination between the fabric and the service devices.