DevNet Associate v1.0

Cisco Compute Management

8.5.1

Cisco Compute Solutions

The Cisco Unified Computing System (UCS), along with its software and SaaS adjuncts, was introduced in 2009 to provide a complete physical and logical plant for compute, networking, and storage in the modern datacenter.

Here, you will examine UCS more closely, then inter-related management tools and services: UCS-Manager, UCS-Director, and the SaaS global infrastructure management system, Intersight.

An overview of UCS

Cisco UCS is engineered to improve flexibility and manageability for conventional infrastructure:

- Resolve problems that prevent, hinder, or make unaffordable the provisioning of optimal compute/storage/network configurations for diverse and dynamic applications.
- Reduce hardware limitations on the efficiency of virtualization, cloud computing, PaaS, container orchestration, and other frameworks that abstract compute, storage, and network resources.
- Enable end-users to provision arbitrary configurations of virtualized infrastructure that map efficiently to underlying physical infrastructure capabilities, capacity, and configuration.
- Enable management of physical infrastructure using the same type of software-centric, "infrastructure as code" disciplines that DevOps is applying, higher in the stack, to accelerate and scale operations.

UCS is "hyperconverged" infrastructure. In very abstract terms, UCS virtualizes physical infrastructure and makes it uniformly software-definable:

- The underlying physical plant provides modular compute and flexible storage capability that connects using an any-to-any switch fabric, making it easy to add capacity.
- Above this, novel firmware and embedded management software abstract the physical layer, making what are conventionally hard-coded characteristics (such as server UUIDs, MAC addresses, disk-drive layouts, and BIOS settings) configurable at boot times, under control of management software. This eliminates the need to manually implement complex hardware configurations and enables a uniform, software-driven, hands-off infrastructure management experience.
- Atop the hyperconverged infrastructure, UCS system management software and services enable web, CLI, API, and tool-based automated management (for example, using Ansible) of up to tens of

thousands of physical servers, as well as the ability to manage a host of non-Cisco infrastructure products.

8.5.2

Cisco UCS Manager

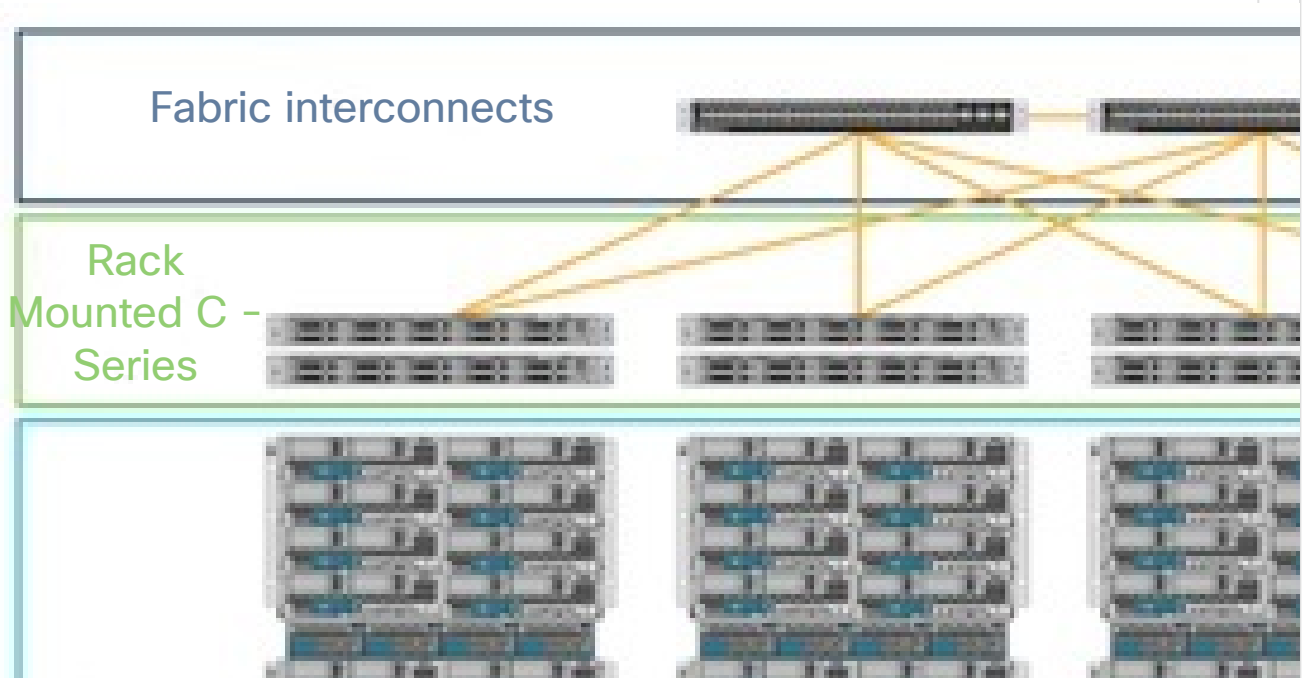
The Cisco Unified Computing System (UCS) is a data center server computer product line composed of computing hardware, switching fabric, embedded management software, and virtualization support.

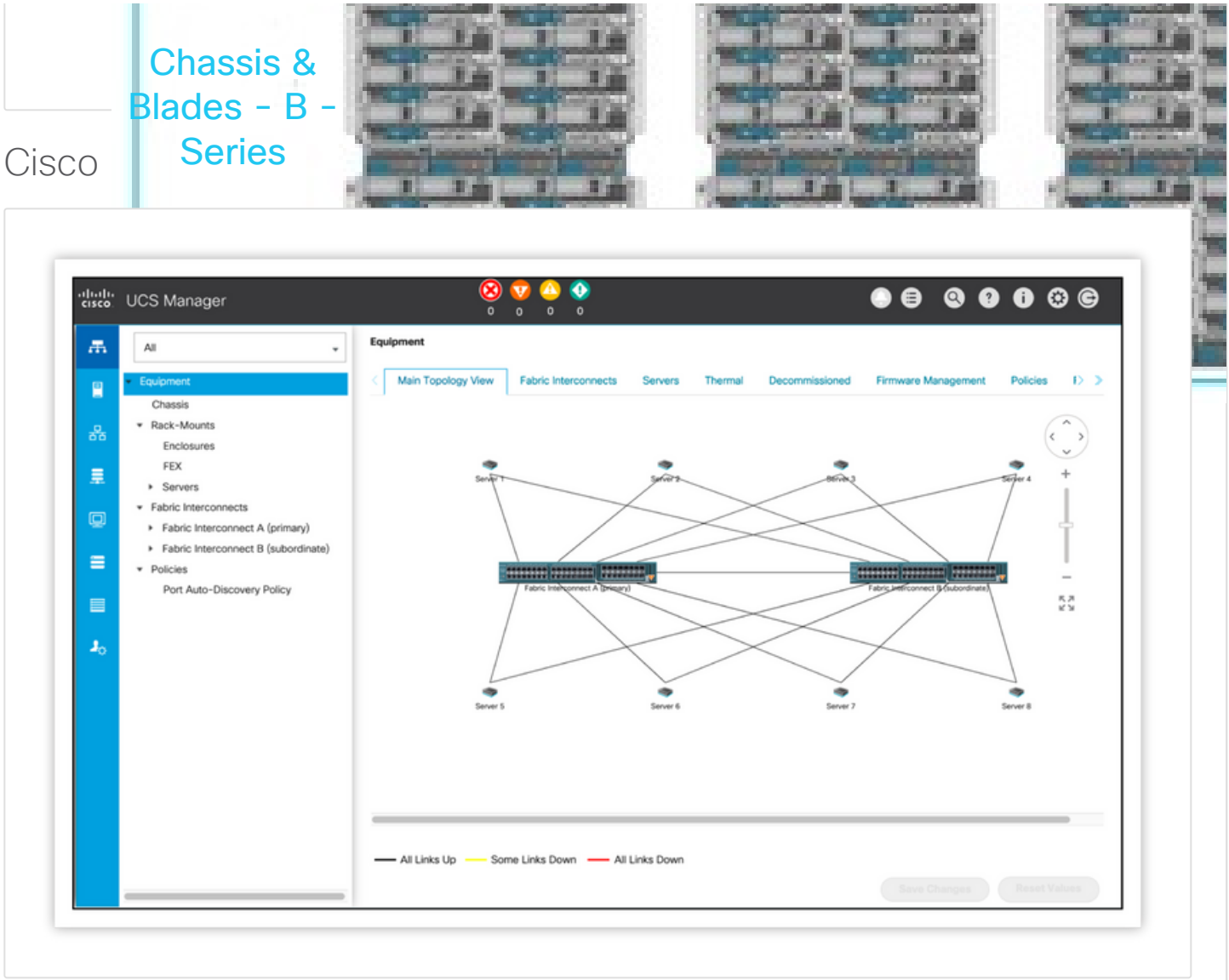
Cisco's definition for Unified Computing is "Unified Computing unifies network virtualization, storage virtualization, and server virtualization into one, within open industry standard technologies and with the network as the platform."

The products provide scalability by integrating many components of a data center. This lets users manage them as a single unit through UCS Manager, UCS Central, and the Cisco Integrated Management Controller. The different management products directly relate to the number of servers being managed:

- Cisco Integrated Management Controller (CIMC) can manage a single physical server.
- Cisco UCS Manager (UCSM) can manage up to 160 physical servers.
- Cisco UCS Central (UCSC) can manage up to 10,000 physical servers.

Cisco UCS Manager Server Overview





A Cisco UCS system or domain can consist of up to two Cisco UCS fabric interconnects and a minimum of one Cisco chassis with one blade or rack-mounted server. Up to 40 chassis with a mixture of blade types and rack-mounted servers can be connected and controlled by a single Cisco UCS Manager.

Cisco UCS Manager runs on the primary fabric interconnect and is assigned a virtual IP address (VIP), with failover capability to the subordinate fabric interconnect. In the event of a failover, the virtual IP address will connect to the subordinate fabric interconnect, making it the new primary fabric interconnect.

The Cisco UCS Manager management requests from the GUI or CLI are encoded in XML. All XML requests to Cisco UCS are asynchronous and terminate on the active Cisco UCS Manager. Cisco UCS Manager mediates all communication within the system; no direct user access to the Cisco UCS components is required.

Cisco UCS Manager is aware of the current configuration and performs automated device discovery whenever a new resource is installed. After a resource is detected, Cisco UCS Manager adds it and its characteristics to the system inventory. Whichever type of management is chosen, the management software is manipulating the same type of structure to manage the server settings. This is the Management Information Model (MIM).

Applying abstraction to servers

Cisco UCS started as a concept to abstract hardware identifiers and properties from physical computing devices, as well as component configurations and settings.

Cisco UCS Management software provides an abstraction layer between server component interfaces and the administrator. This abstraction layer gives the administrator a uniform experience and application methodology for managing server components. The abstraction layer is presented as a web-based GUI, an SSH CLI, and API.

A single server usually has some or all of the following characteristics:

- Many physical disk drives in multiple layouts, supporting a variety of redundancies, security protocols and standards
- Several logical configurations related to boot disk, network access, management interfaces, protocols and standards
- Many hours required to plan, configure and deploy to a data center

Server identifiers and properties are typically "burned-in" or configurable during the server boot phase with a break key sequence. "Burned-in" identifiers, like the server's Universally Unique IDentifier (UUID) and Media Access Control (MAC) addresses are usually assigned to the server motherboard and network adapters at manufacturing. Component configurations and settings, such as disk-drive layout and boot drive assignment, are configurable during the server boot process by using a key sequence to "break-in" to the boot process. The point in the process where an administrator can "break-in" is usually right before the server would need to recognize a specific resource or configuration, and solidify that component or configuration so that subsequent processes can rely upon its state.

Users generally think of servers as the point where an operating system is running. Often, the first thing a user wants to know is what version of OS the server is running. This will indicate whether it is a hypervisor system capable of running multiple virtual machines each with its own guest operating system, or a bare metal server running a single instance of a Linux or Windows operating system. The power of UCS is what happens before the operating system is loaded.

Typically, each component of a server has an individual management interface. The method of accessing the component interface and performing that configuration varies from component to component. For example, a key break during server startup might give access to BIOS settings or disk configuration, while a USB drive with adapter software may need to be inserted into the server to setup network and storage interfaces.

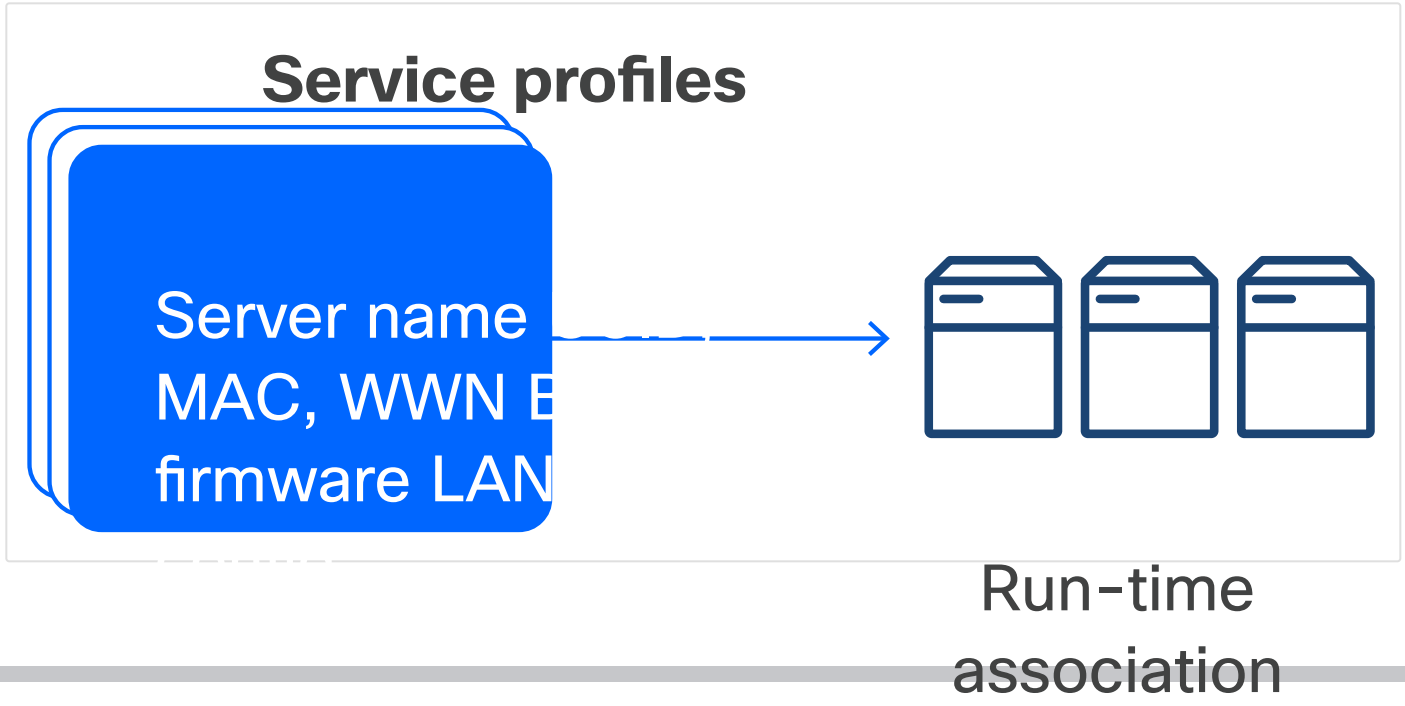
Cisco UCS Manager was the first server management methodology to provide complete control over server component configuration with a single structure called a **Service Profile**. By moving all the server configuration into a Service Profile, UCS Manager enables the administrators to define server identity, addresses, configuration, and settings without actually having a physical server. Cisco-designed hardware enables the overwriting or reassignment of identifiers and addresses, even ones that were "burned-in". In addition, UCS Manager provides address pools and policy definitions that can be consumed by Service Profiles. UCS Manager also provides templates for Service Profiles and templates for network and storage adapters.

Cisco UCS Service Profiles

Cisco UCS Manager uses Service Profiles to assign identity to a server when a Service Profile is associated with a server. UCS Manager can have hundreds of instances of Service Profiles, each with unique addresses and identifiers while also using common policies to manage BIOS settings, boot order, and disk layout.

Each UCS server can only have a single Service Profile association at a time, but Service Profiles can be disassociated from one server and associated to another, thereby applying to the new server the identity that had been applied to the previous server.

Service Profile Application to Physical Servers



8.5.3

Cisco UCS Manager Servers

Cisco UCS Manager is an advancement in server management and deployment for several reasons:

- **Embedded management –** UCS does not require a separate management server because it has an embedded management processor that has global visibility on all the elements that constitute a UCS. This guarantees coordinated control and provides integrated management and diagnostics.
- **Unified Fabric –** UCS is the first server completely designed around the concept of Unified Fabric. Different applications have different I/O requirements, and without Unified Fabric, it is difficult to move applications from one server to another while also preserving the appropriate I/O requirement. Unified Fabric covers LAN, SAN, and the management network.
- **Converged Management –** Cisco UCS Manager manages servers themselves and the way the servers connect to the rest of the data center, using policies for uplink port-channeling, network failover, port management, virtual LAN and SAN connections, and more.

Cisco UCS Manager servers

Cisco UCS Manager servers are either blades that reside in chassis (B-Series Servers) or rack-mounted (C-Series servers). Both are connected to a redundant pair of switches are called UCS Fabric Interconnects (FIs). The FIs provide redundant network and storage paths, redundant management, and a single point of management control for an administrator. When FIs are connected to each other, they are automatically in a primary-subordinate relationship where the primary handles all of the management and data.

When the servers in a UCS system are configured to take advantage of all the redundant capabilities, there should not be a single point of failure. UCS servers can have redundancy, management, networking, storage, power, and cooling.

UCS Manager also keeps track of events, alerts, errors, and statistics. It is able to report these items using various industry-standard management protocols like SNMP and Syslog.

Capabilities, benefits, and use cases

Cisco UCS is deployed in a wide range of industries and provides high performance computing and virtual desktop services to machine learning and artificial intelligence. UCS servers are available in multiple models to target specific workloads but can be configured to accommodate any workload, from bare-metal OS installation to running hundreds of virtual machines.

Cisco UCS Manager supports the entire Cisco UCS server portfolio. It enables server, fabric, and storage provisioning, as well as device discovery, inventory, configuration, diagnostics, monitoring, fault detection, auditing, and statistics collection. You can extend the benefits of Cisco UCS Manager globally across an enterprise to thousands of servers in multiple domains with Cisco UCS Central.

Cisco UCS Manager treats infrastructure as code. It extends the functionality of existing management tools through a broad, mature partner ecosystem. IT organizations can transition to DevOps by evolving existing staff, skills, tools, and processes and making them more efficient.

An open API facilitates integration of Cisco UCS Manager with a wide variety of monitoring, analysis, configuration, deployment, and orchestration tools from other independent software vendors. The API also facilitates customer development through the Cisco UCS PowerTool for PowerShell and a Python SDK.

Cisco UCS Manager includes the following features:

- Supports Cisco UCS B-Series Blade and C-Series Rack Servers, the C3260 storage server, Cisco UCS Mini, and the Cisco HyperFlex
- Programmatically controls server, network, and storage resources, with a unified, policy-driven management, so they can be efficiently managed at scale through software
- Works with HTML 5, Java, or CLI graphical user interfaces
- Can automatically detect, inventory, manage, and provision system components that are added or changed
- Facilitates integration with third-party systems management tools
- Builds on existing skills and supports collaboration across disciplines through role-based administration

Benefits

Based on an analysis of more than 160 case studies, customers on average achieved the following benefits:

- 83 percent decrease in provisioning time
- 75 percent reduction in project implementation time
- 66 percent lower ongoing administrative and management costs

Cisco UCS Unified API

The Cisco UCS Unified API is called unified because the same API methodology is used for the CIMC, Cisco UCS Manager, and Cisco UCS Central. The information that follows pertains mostly to the Cisco UCS Manager XML API, however the concepts apply to CIMC and Cisco UCS Central.

Cisco UCS Management Information Model

All the physical and logical components that comprise Cisco UCS are represented in a hierarchical management information model (MIM), also referred to as the MIT. The MIT is a tree structure with nodes. Each node in the tree represents a managed object (MO) or group of objects that contains the nodes' administrative state and their operational state.

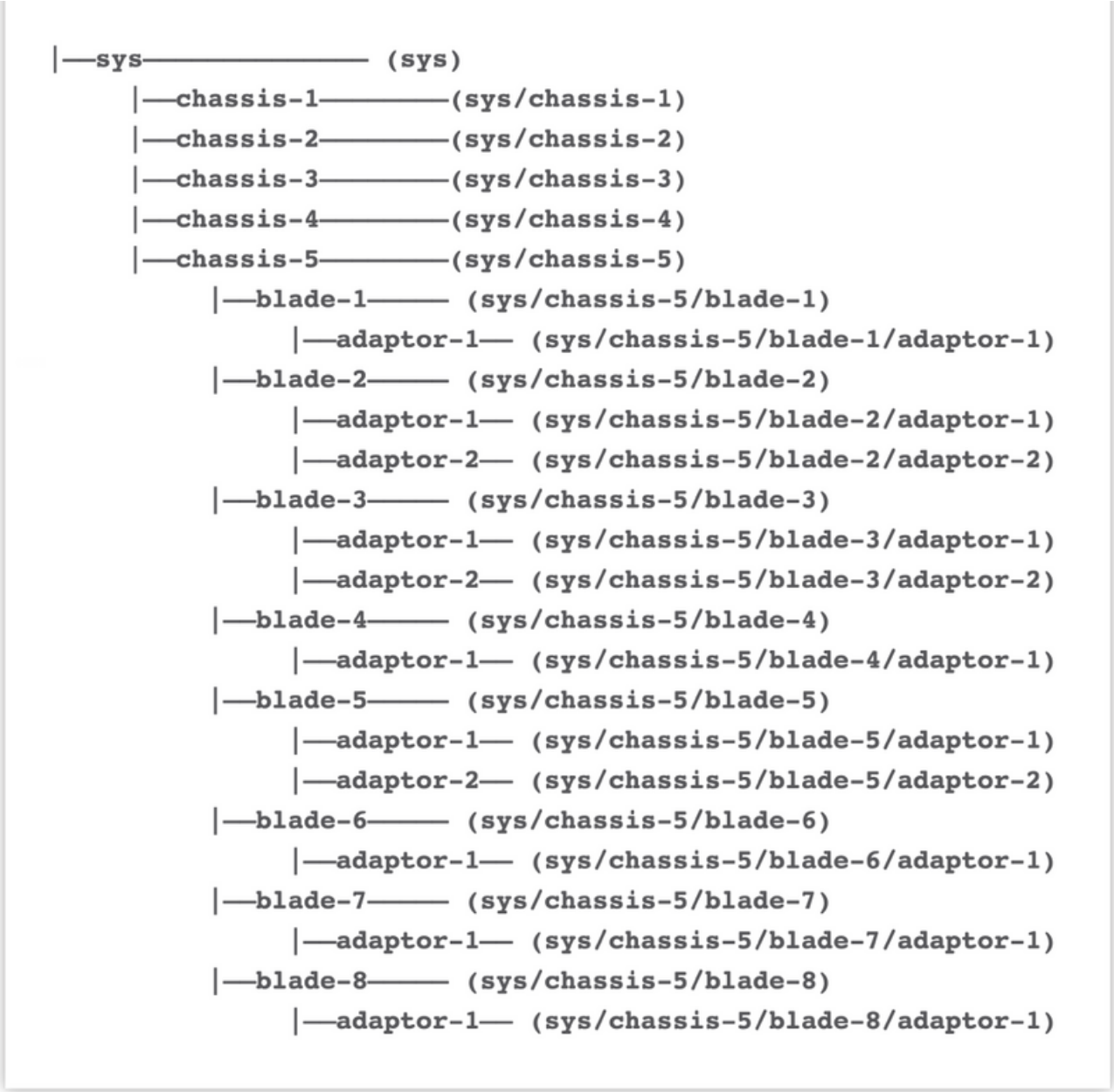
The hierarchical structure starts at the top (`sys`) and contains parent and child nodes. Each node in this tree is a managed object and each object in Cisco UCS has a unique distinguished name (DN) that describes the object and its place in the tree. Managed objects are abstractions of the Cisco UCS resources, such as fabric interconnects, chassis, blades, and rack-mounted servers.

Configuration policies are the majority of the policies in the system and describe the configurations of different Cisco UCS components. Policies determine how the system behaves under specific circumstances. Certain managed objects are not created by users, but are automatically created by the Cisco UCS, for example, power supply objects and fan objects. Invoking the UCS XML API is reading and writing objects to the MIM.

The information model is centrally stored and managed by the Data Management Engine (DME), a process running on the fabric interconnects. When a user initiates an administrative change to a Cisco UCS component (for example, associating a service profile to a server), the DME first applies that change to the information model, and then applies the change to the actual managed endpoint. This approach is called a model-driven framework.

The following branch diagram starts at `sys` from the `topRoot` of the Cisco UCS MIT. The diagram shows a hierarchy that consists of five populated chassis with eight blades in each chassis. All the blades shown have one or more adapters. For simplicity, only chassis number five is expanded in the graphic below.

Cisco UCS MIT Hierarchical Structure



8.5.4

Cisco UCS Managed Objects

Cisco UCS Managed Objects are XML representations of a physical and logical entities in the UCS system. A boot policy object named **default** with a single child object that indicates booting from virtual media looks as follows:

```
<lsbootPolicy
  bootMode="legacy"
  childAction="deleteNonPresent"
```

```
  descr="Default Boot Policy"
  dn="org-root/boot-policy-default"
  enforceVnicName="no"
  intId="85315"
  name="default"
  policyLevel="0"
  policyOwner="local"
  propAcl="0"
  purpose="operational"
  rebootOnUpdate="no">
    <lsbootVirtualMedia
      access="read-write"
      childAction="deleteNonPresent"
      lunId="unspecified"
      mappingName=""
      order="4"
      propAcl="0"
      rn="read-write-vm"
      type="virtual-media"/>
</lsbootPolicy>
```

The class name is the XML element, in this case, `lsbootPolicy` for the boot policy object, followed by a number of attributes and potentially containing children objects as shown here by the `lsbootVirtualMedia` object. The `lsbootVirtualMedia` object belongs to the class `lsbootVirtualMedia`.

A UCS managed object can have many attributes and can contain many children, the children objects can also contain many children, and so on.

Object naming

You can identify a specific object by its Distinguished Name or by its Relative Name.

The Distinguished Name (DN) enables you to unambiguously identify a target object. All UCS managed objects have a unique DN; no other object in the entire UCS management domain can have the same DN.

The DN has the following format consisting of a series of RNs:

`dn = {rn}/{rn}/{rn}/{rn}...`

In the following example, the DN provides a fully qualified path for adaptor-1 from the top of the object tree to the object. The DN specifies the exact managed object on which the API call is operating.

```
<dn="sys/chassis-5/blade-2/adaptor-1"/>
```

The Relative Name (RN) identifies an object within the context of its parent object. The DN is composed of a sequence of RNs. For example, this DN:

```
<dn="sys/chassis-5/blade-2/adaptor-1/host-eth-2"/>
```

is composed of the following RNs:

- `topSystem`: `rn="sys"`
- `equipmentChassis`: `rn="chassis-5"`
- `computeBlade`: `rn ="blade-2"`
- `adaptorUnit`: `rn="adaptor-1"`
- `adaptorHostEthIf`: `rn="host-eth-2"`

Object classes

All managed objects belong to a class indicating the type of UCS resource the object represents. For example, `computeBlade` is the class that all UCS blades belong to, and `fabricVlan` is the class for all VLAN objects.

UCS XML API

The Cisco UCS Manager XML API, like other APIs, provides methods to authenticate, query, and configure.

Authentication methods

Authentication methods authenticate and maintain an API session with UCS Manager:

- `aaaLogin` - Initial method for logging in and retrieving an authentication cookie.
- `aaaRefresh` - Refreshes the current authentication cookie.
- `aaaLogout` - Exits the current session and deactivates the authentication cookie.

Operations are performed using the HTTP post method over TCP. Cisco UCS supports both HTTP and HTTPS requests and HTTP and HTTPS can be configured to use different port numbers, but TCP/443 (or TCP/80 for non-secure connections) is used by default. The HTTP POST body contains the XML configuration.

Query methods

Query methods obtain information on the current configuration state of an object. The following are query examples:

- `configResolveDn` - Retrieves objects by DN.
- `configResolveDns` - Retrieves objects by a set of DNs.
- `configResolveClass` - Retrieves objects of a given class.
- `configResolveClasses` - Retrieves objects of multiple classes.
- `configFindDnsByClassId` - Retrieves the DNs of a specified class.
- `configResolveChildren` - Retrieves the child objects of an object.
- `configResolveParent` - Retrieves the parent object of an object.
- `configScope—Performs` – Performs class queries on a DN in the MIT.

Query filters

The API also provides a set of query filters. These filters can be passed as part of a query and are used to identify the desired result set.

Simple filters

There are two simple filters, the true filter and false filter:

- `True` - Objects with the Boolean condition of true.
- `False` - Objects with the Boolean condition of false.

Property filters

The property filters use the values of an object's properties as the criteria for inclusion in a result set. To create most property filters, `classId` and `propertyId` of the target object or property is required, along with a value for comparison.

- `Equality` - Objects with the identified property of "equal" to the provided property value.
- `Not equal` - Objects with the identified property of "not equal" to the provided property value.
- `Greater than` - Objects with the identified property of "greater than" the provided property value.
- `Greater than or equal` - Objects with the identified property of "is greater than or equal" to the provided property value.
- `Less than` - Objects with the identified property of "less than" the provided property value.
- `Less than or equal` - Objects with the identified property of "less than or equal" to the provided property value.
- `Wildcard` - Objects where the identified property includes a wildcard. Supported wildcards include "%" or "*" (any sequence of characters), "?" or "-" (any single character).

Composite filters

The composite filters are composed of two or more component filters. They enable greater flexibility in creating result sets. For example, a composite filter could restrict the result set to only those objects that were accepted by at least one of the contained filters.

- `AND` - Result set must pass the filtering criteria of each component filter. For example, to obtain all compute blades with `totalMemory` greater than 64 megabytes and operability of operable, the filter is composed of one "greater than" filter and one "equality" filter.
- `OR` - Result set must pass the filtering criteria of at least one of the component filters. For example, to obtain all the service profiles that have an `assignmentState` of unassigned or an association state value of unassociated, the filter is composed of two "equality" filters.
- `Between` - Result set is those objects that fall between the range of the first specified value and second specified value, inclusive. For example, all faults that occurred starting on the first date and ending on the last date.
- `XOR` - Result set is those objects that pass the filtering criteria of no more than one of the composite's component filters.

Modifier filter

A modifier filter changes the results of a contained filter. The only modifier filter that is currently supported is the `NOT` filter, which negates the result of a contained filter. Use this filter to obtain objects that do not match contained criteria.

Configuration methods

There are several methods to make configuration changes to managed objects. These changes can be applied to the whole object tree, a subtree rooted at a specified object, or an individual object. The following are examples of configuration methods:

- `configConfMo` – Affects a single managed object.
- `configConfMos` – Affects multiple subtrees.
- `configConfMoGroup` – Applies the same configuration changes to multiple subtree structures or managed objects.

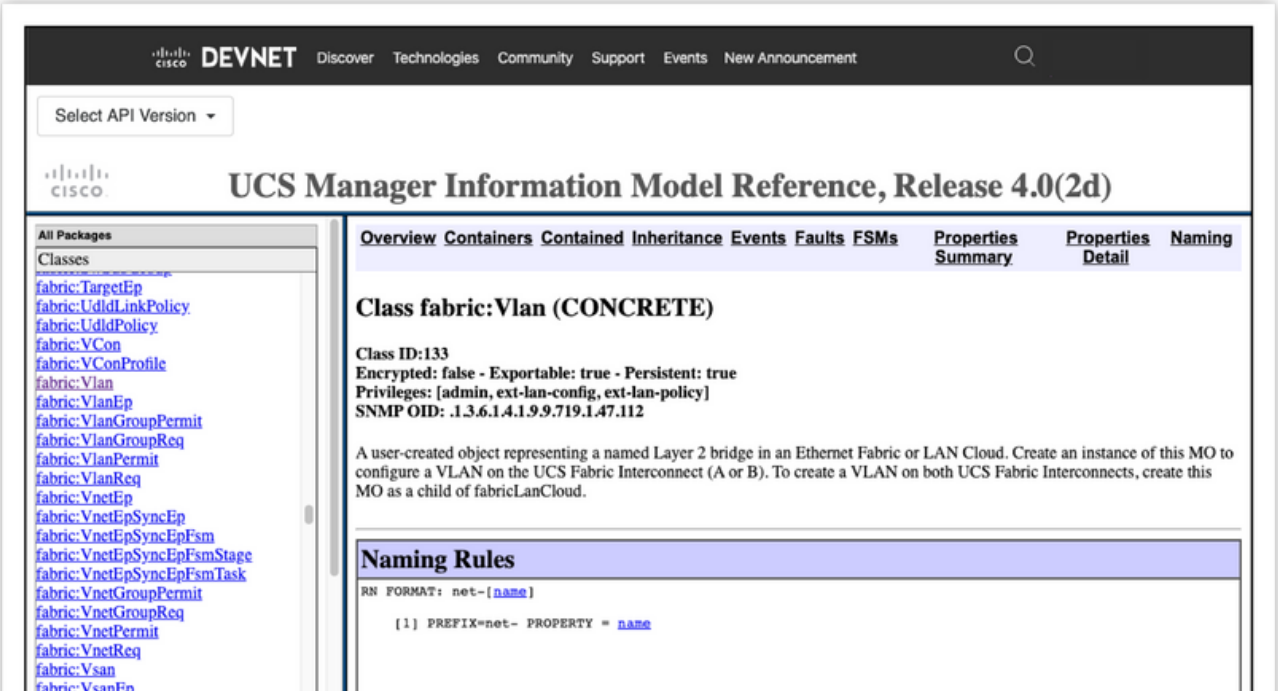
Cisco UCS API documentation

Cisco UCS API documentation is typically referred to as the UCS Object Model documentation. The Object Model documentation is available with the UCS Platform Emulator or online.

Every UCS object class and method is listed in the documentation along with UCS types, events, faults, errors, and Syslog messages. The documentation is a wealth of information providing information about every aspect of a UCS managed object.

As an example of how the documentation works, you can explore the `fabricVlan` object.

UCS Object Model Documentation – Overview and Naming Rules



8.5.5

Cisco UCS Manager Documentation

The left navigation menu enables you to select an object. The right pane displays the object information divided into sections related to various aspects of the object.

Overview in documentation

The Overview section indicates whether the object is Abstract or Concrete. All objects that represent a physical or logical entity are Concrete. Abstract objects are used for inheritance where several different objects can inherit shared attributes from a base object. Abstract objects can also represent an aggregation of objects. For example, `computeServer` is an Abstract object that represents an aggregation of the `computeBlade` object and `computeRackUnit` object.

The Overview also contains information related to encryption, privileges, the SNMP OID and a description of the object.

Naming Rules

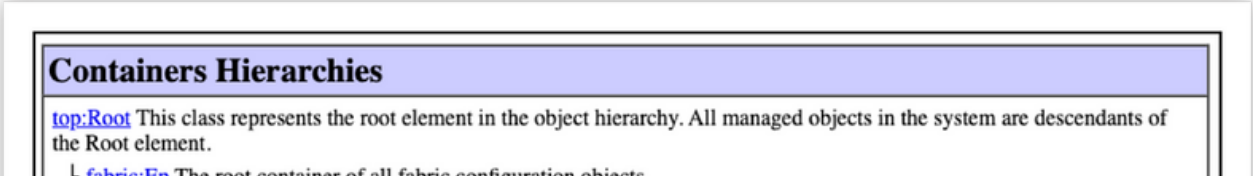
The Naming Rules indicate the object prefix. In the case of a `fabricVlan` if the object prefix is `net-` that means a VLAN object defined in UCS will have `net-` in front of the provided VLAN name. For example, if a VLAN is created with the name `backupvlan` the name portion of the object DN will be `net-backupvlan`

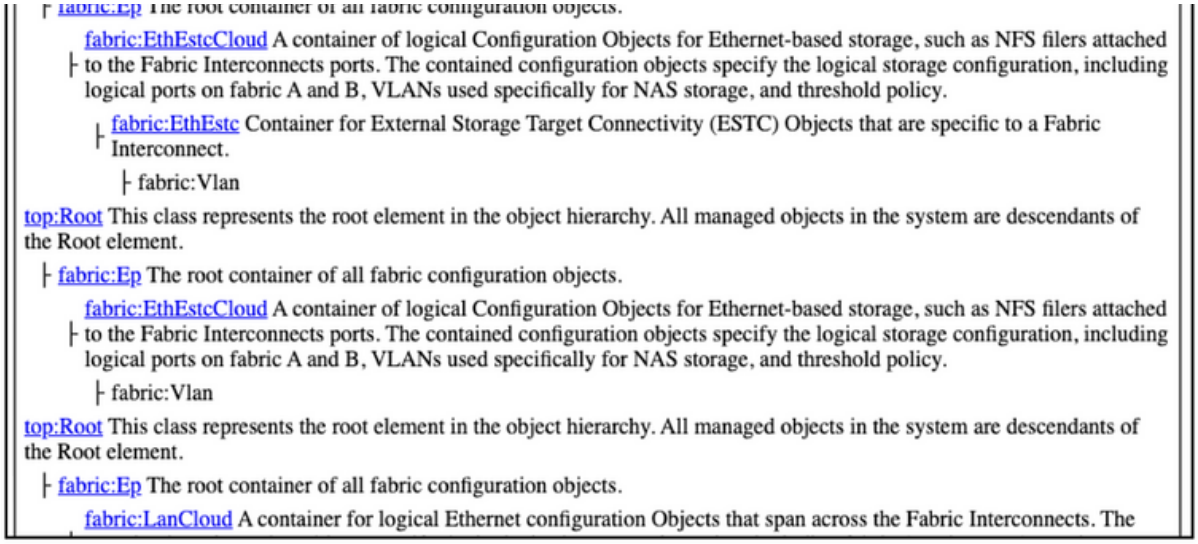
The Naming Rules also define the different DNs that can be assigned to a `fabricVlan` object. This is because UCS VLANs can be different types of VLANs and reside in different parts of the MIM.

Containers Hierarchies

Containers Hierarchies display where the `fabricVlan` object can be contained, in other words, where the object can reside in the MIM.

UCS Object Model Documentation – Containers Hierarchies

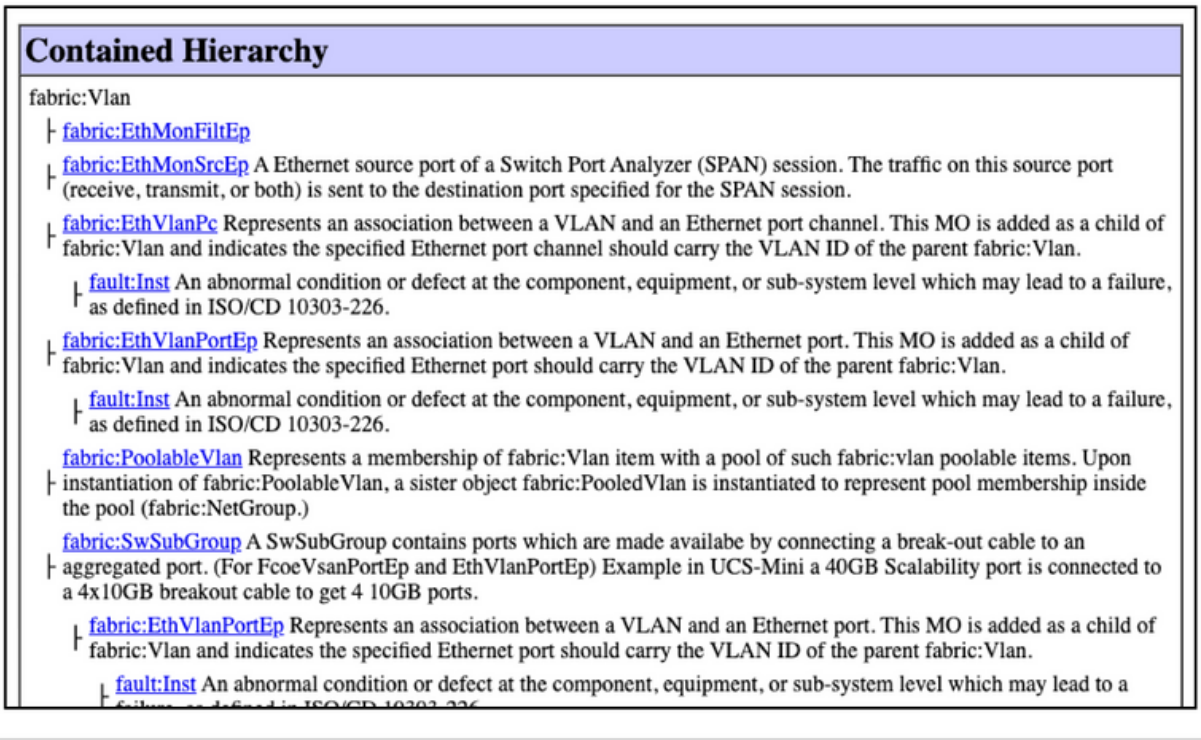




Contained Hierarchy

Contained Hierarchy displays what objects the `fabricVlan` can contain and what objects those contained objects can contain, and so on.

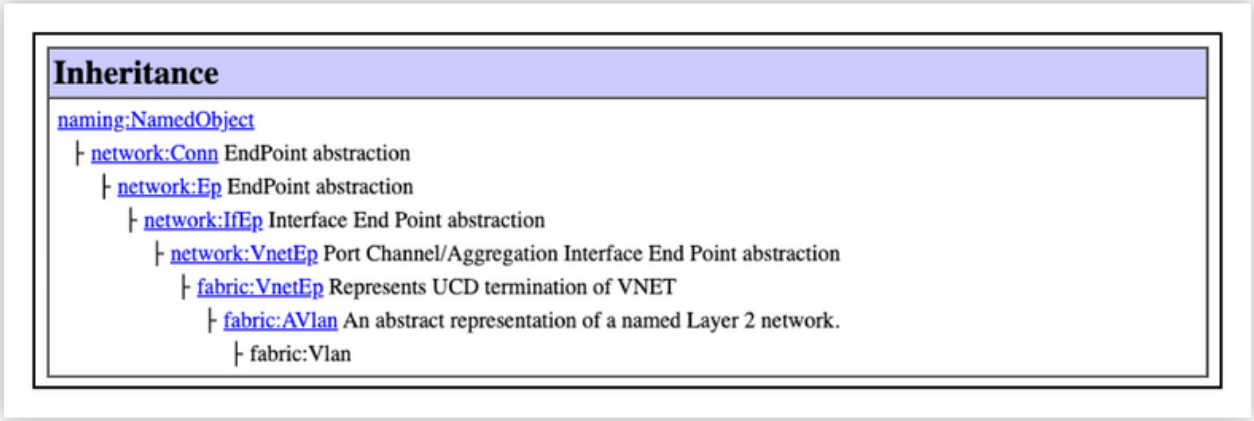
UCS Object Model Documentation – Contained Hierarchy



Inheritance

Inheritance displays all the objects prior to the `fabricVlan` object that the `fabricVlan` inherited attributes from.

UCS Object Model Documentation – Inheritance



Inheritance

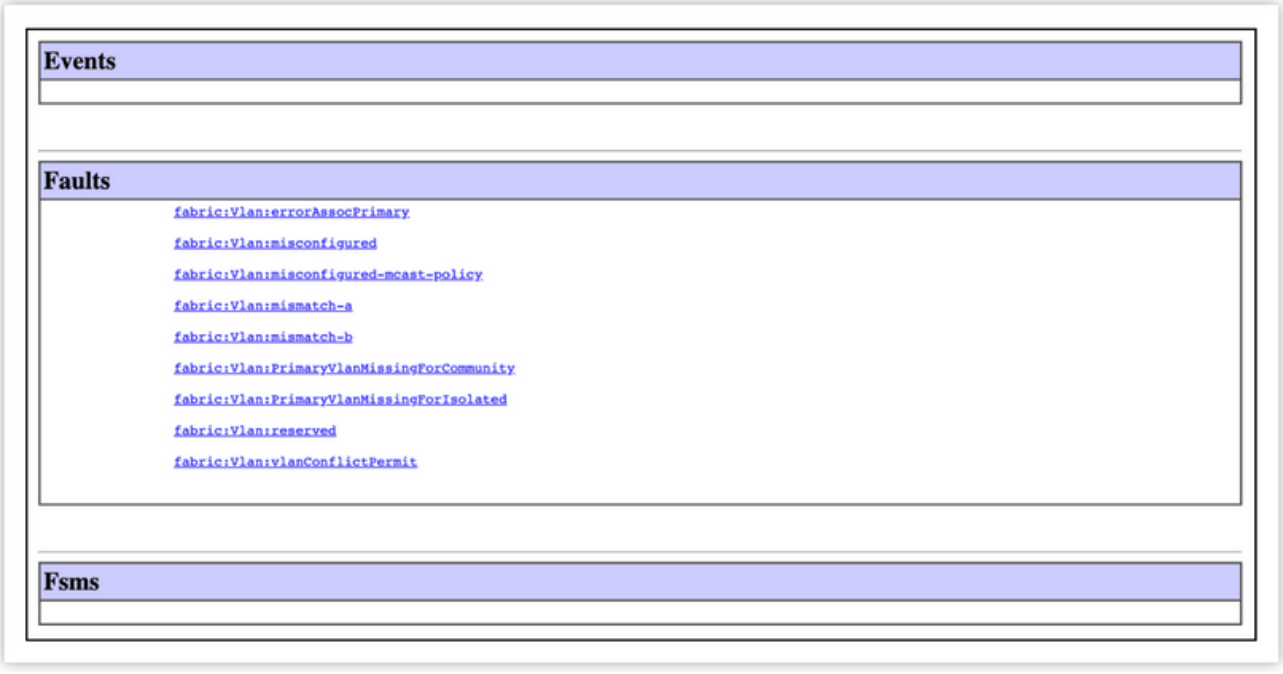
naming:NamedObject
- network:Conn EndPoint abstraction
  - network:Ep EndPoint abstraction
    - network:IfEp Interface End Point abstraction
      - network:VnetEp Port Channel/Aggregation Interface End Point abstraction
        - fabric:VnetEp Represents UCD termination of VNET
          - fabric:AVlan An abstract representation of a named Layer 2 network.
            - fabric:Vlan

Events, faults, and FSMs

Objects can be changed by an administrator or a process. Mutations or changes to the object are events. Mutations happen as the object is created, modified, and deleted, and may generate events and faults. The process of mutating is performed by one or more Finite State Machine processes (FSMs).

Events, faults, and FSMs are also objects and are attached to the parent object for which they were generated. The fabricVlan object does not have events and associated FSMs.

UCS Object Model Documentation – Events, Faults, and FSMs



Events

Faults

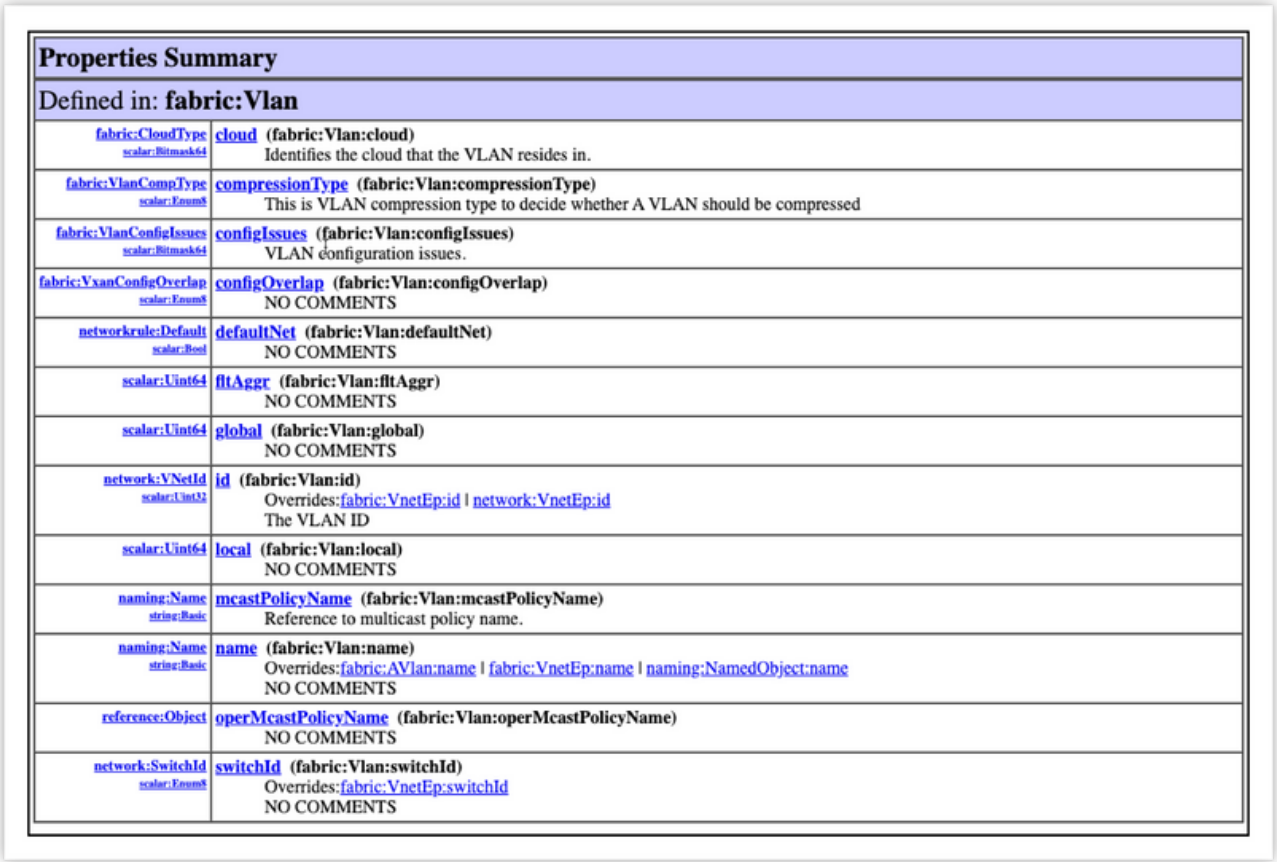
fabric:Vlan:errorAssocPrimary
fabric:Vlan:misconfigured
fabric:Vlan:misconfigured-mcast-policy
fabric:Vlan:mismatch-a
fabric:Vlan:mismatch-b
fabric:Vlan:PrimaryVlanMissingForCommunity
fabric:Vlan:PrimaryVlanMissingForIsolated
fabric:Vlan:reserved
fabric:Vlan:vlanConflictPermit

Fsms

Properties Summary

The Properties Summary lists all the properties or attributes of the object. Each property's name and type are displayed. Properties are grouped by the object where the property was first defined. Note that objects can inherit properties from another higher level object definition.

UCS Object Model Documentation – Properties Summary



Properties Summary

Defined in: **fabric:Vlan**

fabric:CloudType scalar:Bitmask64	**cloud** (fabric:Vlan:cloud) Identifies the cloud that the VLAN resides in.
fabric:VlanCompType scalar:Enum8	**compressionType** (fabric:Vlan:compressionType) This is VLAN compression type to decide whether A VLAN should be compressed
fabric:VlanConfigIssues scalar:Bitmask64	**configIssues** (fabric:Vlan:configIssues) VLAN configuration issues.
fabric:VxanConfigOverlap scalar:Enum8	**configOverlap** (fabric:Vlan:configOverlap) NO COMMENTS
networkrule:Default scalar:Bool	**defaultNet** (fabric:Vlan:defaultNet) NO COMMENTS
scalar:Uint64	**fltAggr** (fabric:Vlan:fltAggr) NO COMMENTS
scalar:Uint64	**global** (fabric:Vlan:global) NO COMMENTS
network:VNetId scalar:Uint32	**id** (fabric:Vlan:id) Overrides:fabric:VnetEp:id | network:VnetEp:id The VLAN ID
scalar:Uint64	**local** (fabric:Vlan:local) NO COMMENTS
naming:Name string:Basic	**mcastPolicyName** (fabric:Vlan:mcastPolicyName) Reference to multicast policy name.
naming:Name string:Basic	**name** (fabric:Vlan:name) Overrides:fabric:AVlan:name | fabric:VnetEp:name | naming:NamedObject:name NO COMMENTS
reference:Object	**operMcastPolicyName** (fabric:Vlan:operMcastPolicyName) NO COMMENTS
network:SwitchId scalar:Enum8	**switchId** (fabric:Vlan:switchId) Overrides:fabric:VnetEp:switchId NO COMMENTS

Properties Detail

Each object property is defined completely in the Properties Detail. The `fabricVlan` name property has details for type, encryption, access, and more.

Note the Property Validators. The validator for range indicates that the `fabricVlan` name property must be at least one character long but not greater than thirty-two characters. The validator for what characters are allowed is shown as a regular expression: `a-zA-Z0-9_.:-]+`

This regular expression indicates that any combination of the characters between the brackets is allowed in a fabricVlan name.

UCS Object Model Documentation – Properties Summary

Properties Detail

name

Type: naming:Name
Primitive Type: string:Basic

Overrides:fabric:AVlan:name | fabric:VnetEp:name | naming:NamedObject:name

Units: null

Encrypted: false

Naming Property -- [NAMING RULES]

Access: naming

Category: TopLevelRegular

Property Validators:
Range: min: "1" max: "32"
Allowed Chars:
Regex: [a-zA-Z0-9_.:-]+

Comments:

NO COMMENTS

XML, SDKs, tools, and resources

The UCS XML API uses XML as a method to encode requests and responses via the API, and also uses XML to define the entire Object Model. The entirety of XML used to define the MIM is known as the XML schema. All UCS objects are described in the XML schema. The schema defines the objects, their attributes, and the associated values. All components of UCS are always available with the XML API.

There are a wide variety of libraries available for multiple programming languages for manipulating XML and interacting with an HTTP service. An SDK provides several convenience functions and features. It removes a lot of the overhead associated with writing XML directly.

When the Cisco UCS XML API was first released, API interactions were performed by constructing XML documents and managing HTTP interactions from a client application. Those interactions were similar to the following examples.

Authentication request and response

```
<aaaLogin inName="admin" inPassword="password" />
<aaaLogin
  response="yes"
  outCookie="cookie"
  outRefreshPeriod="600"
  outPriv="aaa,ext-lan-policy,ext-lan-qos,ext-san-policy,operations, pod-policy,pod-
qos,read-only"
  outDomains="mgmt02-dummy"
  outChannel="noencssl"
```

```
  outEvtChannel="noencssl">
</aaaLogin>
```

Successful authentication requests return an authentication cookie. The cookie is used for all subsequent requests. The cookie eventually expires and needs to be refreshed. Refreshing the cookie resets the expiration time and returns a new cookie.

Query request and response

```
<configResolveDn
  cookie="cookie"
  inHierarchical="false"
  dn="mac/10:00:00:00:00:03" />
<configResolveDn
  dn="mac/10:00:00:00:00:03"
  cookie="cookie"
  response="yes">
  <outConfig>
    <macpoolAddr assigned="yes"
      assignedToDn="org-root/ls-SP01/ether-eth1"
      dn="mac/10:00:00:00:00:03"
      id="10:00:00:00:00:03"
      owner="pool" />
  </outConfig>
</configResolveDn>
```

Whether making a query or performing a configuration, the XML document needs to be prepared for the request and parsed when XML is returned in the response.

8.5.6

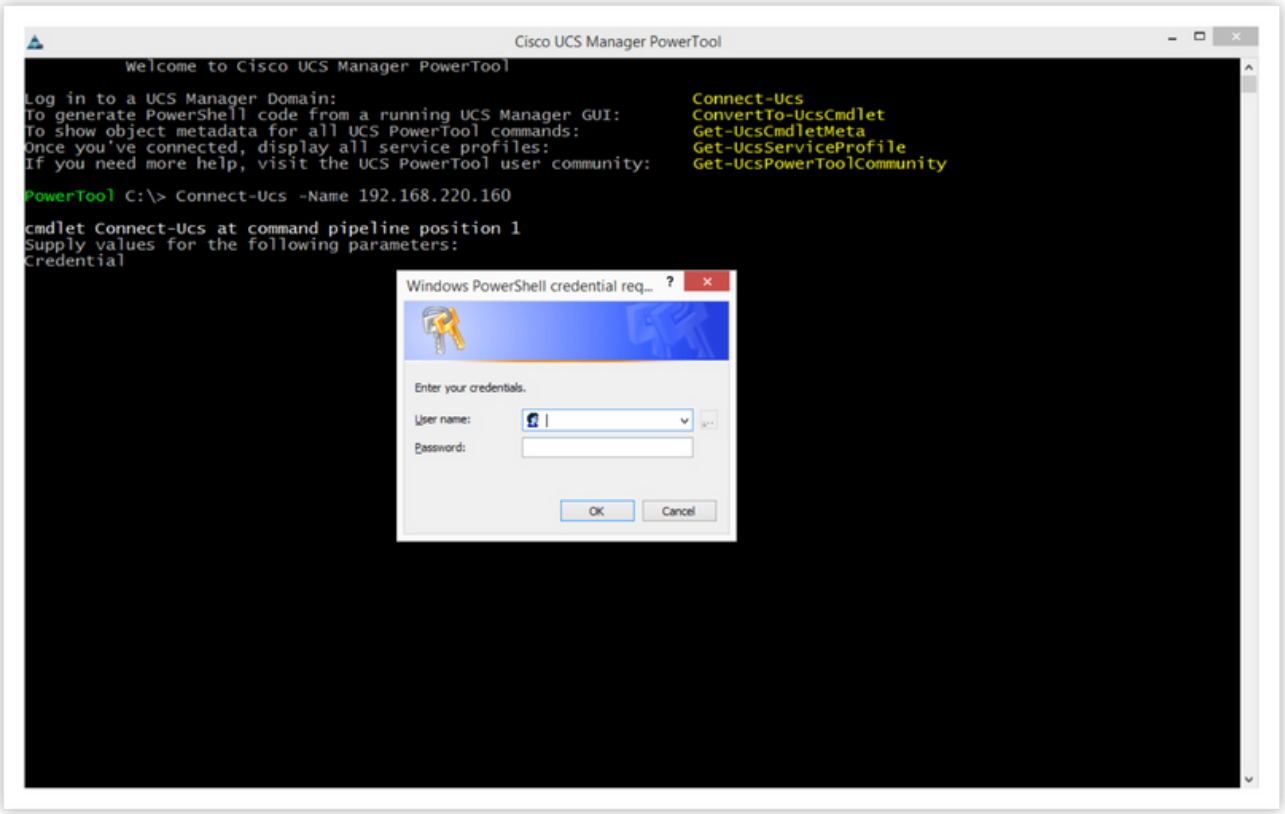
Cisco UCS Power Tool

UCS PowerTool is a library of PowerShell Cmdlets that enable the management of UCS environments from Microsoft Operating Systems, via the UCS XML API. The UCS XML schema is used to generate more than 98% of the UCS PowerTool Library. Using the XML schema to generate the PowerTool, Cmdlets ensure that the Cmdlets are completely aware of objects, their containment, properties and the details associated with each property.

The Cmdlet to authenticate with a UCS Manager is as follows:

```
Connect-Ucs -Name <ip-address-ucs-manager>
```

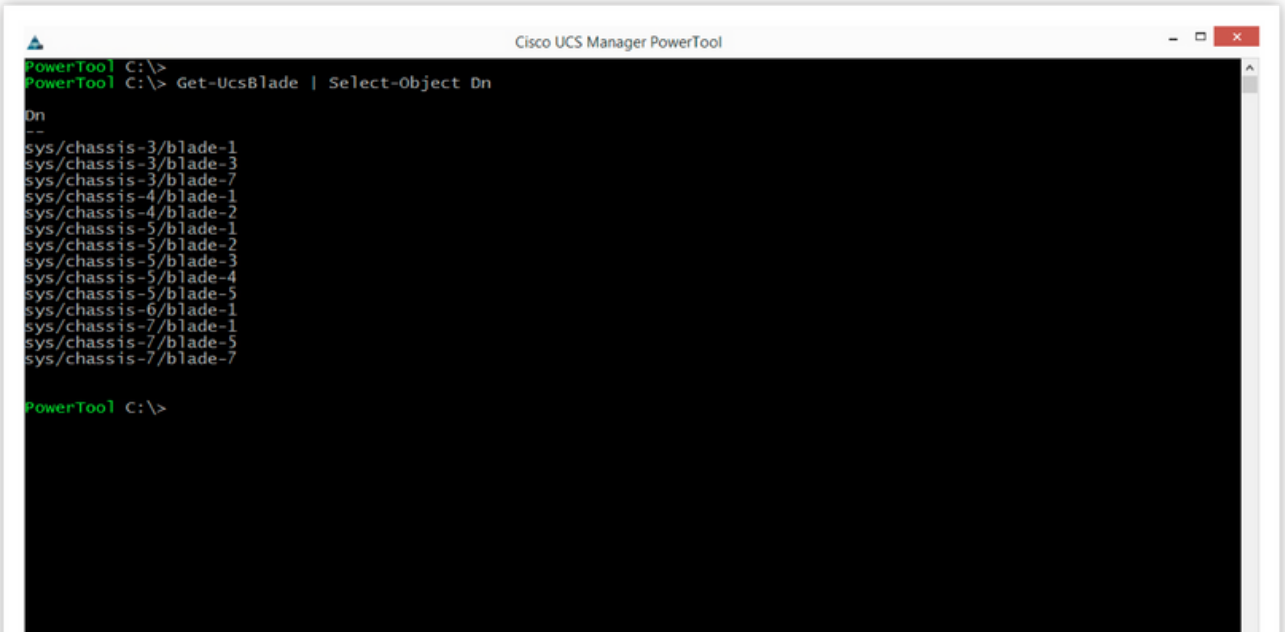
UCS PowerTool - Authentication

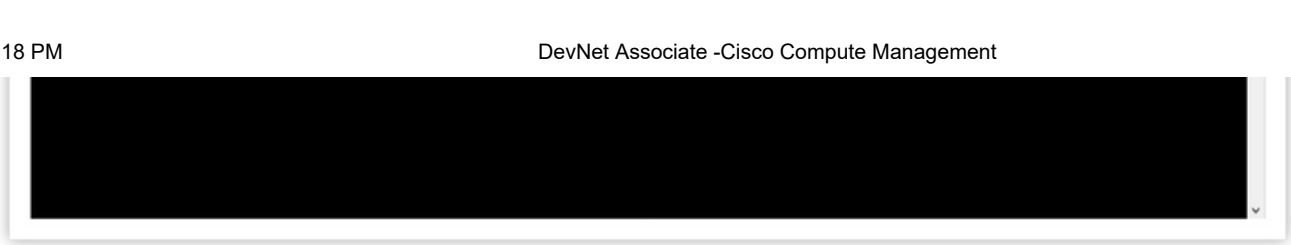


The Cmdlet to query UCS Blades and view their DNs is as follows:

```
Get-UcsBlade | Select-Object Dn
```

UCS PowerTool - Blade Query





The Cmdlet to create UCS VLAN 100 is as follows:

```
Get-UcsLanCloud | Add-UcsVlan -Name vlan100 -Id 100
```

UCS PowerTool OS support

UCS PowerTool is a library of PowerShell Cmdlets. PowerShell Desktop runs on Windows and PowerShell Core runs on Linux variants including macOS. UCS PowerTool and UCS PowerTool Core support those releases of PowerShell.

UCS PowerTool has nearly 6000 Cmdlets to manage every aspect of Cisco UCS Systems.

UCS Python SDK

UCS Python SDK is a set of Python modules, each containing one or more classes, developed specifically to automate UCS Manager via the UCS XML API. The UCS Python SDK is developed with PEP8 compliance and supports every Object in the UCS Object Model. The UCS XML schema is used to generate more than 98% of the UCS Python SDK. Using the XML schema to generate the UCS Python SDK ensures that the Python modules are completely aware of objects, their containment, properties, and the details associated with each property.

Similar to UCS PowerTool, the UCS Python SDK provides hundreds of Python modules to interact with UCS objects.

The UCS Python SDK code to authenticate with a UCS Manager is as follows:

```
from ucsmsdk.ucshandle import UcsHandle
handle = UcsHandle("ucs-manager-ip", "username", "passsword")
handle.login()
```

The UCS Python SDK code to query UCS Blades and view their DNs is as follows:

```
from ucsmsdk.ucshandle import UcsHandle
handle = UcsHandle("ucs-manager-ip", "username", "passsword")
handle.login()
blades = handle.query_classid("computeBlade")
for blade in blades:
  print(blade.dn)
```

The UCS Python SDK to Create UCS VLAN 100 is as follows, the code shown here is PEP8 compliant and will score a 10 out of 10 when run through pylint:

```
from ucsmsdk.ucshandle import UcsHandle
from ucsmsdk.mometa.fabric.FabricVlan import FabricVlan
# Create a Login Handle and Login
HANDLE = UcsHandle("ucs-manager-ip", "username", "password")
HANDLE.login()
# Query the FabricLanCloud, under which VLAN Objects are inserted
FABRIC_LAN_CLOUD = HANDLE.query_classid("FabricLanCloud")
# Instantiate a VLAN Object, minimally "id" and "name" are required
VLAN = FabricVlan(
    parent_mo_or_dn=FABRIC_LAN_CLOUD[0],
    name="vlan100",
    id="100"
    )
# Add the instantiated VLAN Object to the HANDLE
HANDLE.add_mo(VLAN)
# Commit the HANDLE to add the VLAN to UCS Manager
HANDLE.commit()
# Logout
HANDLE.logout()
```

UCS Python SDK OS support

The UCS Python SDK is supported wherever Python is supported.

UCS Ansible

UCS Ansible is available for UCS Manager and the Cisco Integrated Management Controller.

Similar to UCS Python SDK, UCS Ansible combines the UCS Manager authentication with the object mutation or object query. This makes a lot of sense because UCS Ansible modules are written using the UCS Python SDK.

UCS Ansible modules are called as tasks in an Ansible playbook. All the features and capabilities of the Ansible Domain Specific language are available to UCS Ansible Modules.

Ansible playbook that queries a UCS Manager for UCS VLANs is shown below:

```
---
- name: UCS Query
  hosts: ucs
  connection: local
  gather_facts: no
  tasks:
  - name: Query UCS
```

```
    ucs_query:
      hostname: "{{ ucs_hostname }}"
      username: "{{ ucs_username }}"
      password: "{{ ucs_password }}"
      class_ids: fabricVlan
    register: response
```

To create, update, and delete multiple UCS Organization use the following Ansible Playbook:

```
---
- hosts: ucs
  connection: local
  gather_facts: no
  tasks:
  - name: Test for a UCS hostname,  username, and password
    fail:
      msg: 'Please define the following variables: ucs_hostname, ucs_username and
ucs_password.'
    when: ucs_hostname is not defined or ucs_username is not defined or ucs_password
is not defined
    vars:
      login_info: &login_info
        hostname: "{{ ucs_hostname }}"
        username: "{{ ucs_username }}"
        password: "{{ ucs_password }}"
  - name: Add UCS Organization
    ucs_org:
      <<: *login_info
      org_name: test
      description: testing org
      state: present
      delegate_to: localhost
  - name: Update UCS Organization
    ucs_org:
      <<: *login_info
      org_name: test
      description: Testing org
      state: present
      delegate_to: localhost
  - name: Add UCS Organization
    ucs_org:
      <<: *login_info
      org_name: level1
      parent_org_path: root
      description: level1 org
      state: present
      delegate_to: localhost
  - name: Add UCS Organization
```

```
    ucs_org:
      <<: *login_info
      org_name: level2
      parent_org_path: root/level1
      description: level2 org
      state: present
      delegate_to: localhost
  - name: Add UCS Organization
    ucs_org:
      <<: *login_info
      org_name: level3
      parent_org_path: root/level1/level2
      description: level3 org
      state: present
      delegate_to: localhost
  - name: Remove UCS Organization
    ucs_org:
      <<: *login_info
      org_name: level2
      parent_org_path: root/level1
      state: absent
      delegate_to: localhost
```

Cisco UCS Manager XML API summary

The Cisco UCS Manager XML API, which is part of the Cisco UCS Unified API, is the foundation for everything in UCS. XML defines the Object Model schema, encodes the object in the MIM, and is used in the API to encode the requests and responses from the UCS platforms.

The PowerShell and Python adaptations for the UCS XML API abstract the developer from the XML and provide a better experience for the developer. These tools manage cookie refreshes and object mutations, provide the ability to wrap interactions in transactions, and create watch methods with callback functions that trigger when a particular setting is applied, or a state is observed.

Ansible abstracts the developer to another level and provides a declarative model that puts the burden on the module to achieve what is being declared.

The UCS XML API provides complete coverage of the objects presented through the UCS management interfaces, enabling any tool or SDK built on top of XML API the ability to control all the objects in the UCS Object model.

8.5.7

Cisco UCS Director

Cisco Unified Computing System (UCS) Director is a complete, highly secure, end-to-end management, orchestration, and automation solution for a wide array of Cisco and non-Cisco data center infrastructure components, and for converged infrastructure solutions based on the UCS and Cisco Nexus platforms.

Cisco UCS Director is a 64-bit appliance that uses the following standard templates:

- Open Virtualization Format (OVF) for VMware vSphere
- Virtual Hard Disk (VHD) for Microsoft Hyper-V

Management through Cisco UCS Director

Cisco UCS Director extends the unification of computing and networking layers through Cisco UCS to provide visibility and management of data center infrastructure components. You can use Cisco UCS Director to configure, administer, and monitor supported Cisco and non-Cisco components. You can use UCS Director to perform the following tasks:

- Create, clone, and deploy service profiles and templates for all Cisco UCS servers and compute applications.
- Monitor organizational usage, trends, and capacity across a converged infrastructure on a continuous basis. For example, you can view heat maps that show virtual machine (VM) utilization across all your data centers.
- Deploy and add capacity to converged infrastructures in a consistent, repeatable manner.
- Manage, monitor, and report on data center components, such as Cisco UCS domains or Cisco Nexus network devices.
- Extend virtual service catalogs to include services for your physical infrastructure.
- Manage secure multi-tenant environments to accommodate virtualized workloads that run with non-virtualized workloads.

Automation and orchestration with Cisco UCS Director

Cisco UCS Director enables you to build workflows that provide automation services and to publish the workflows and extend their services to your users on demand. You can build Cisco UCS Director workflows to automate simple or complex provisioning and configuration processes.

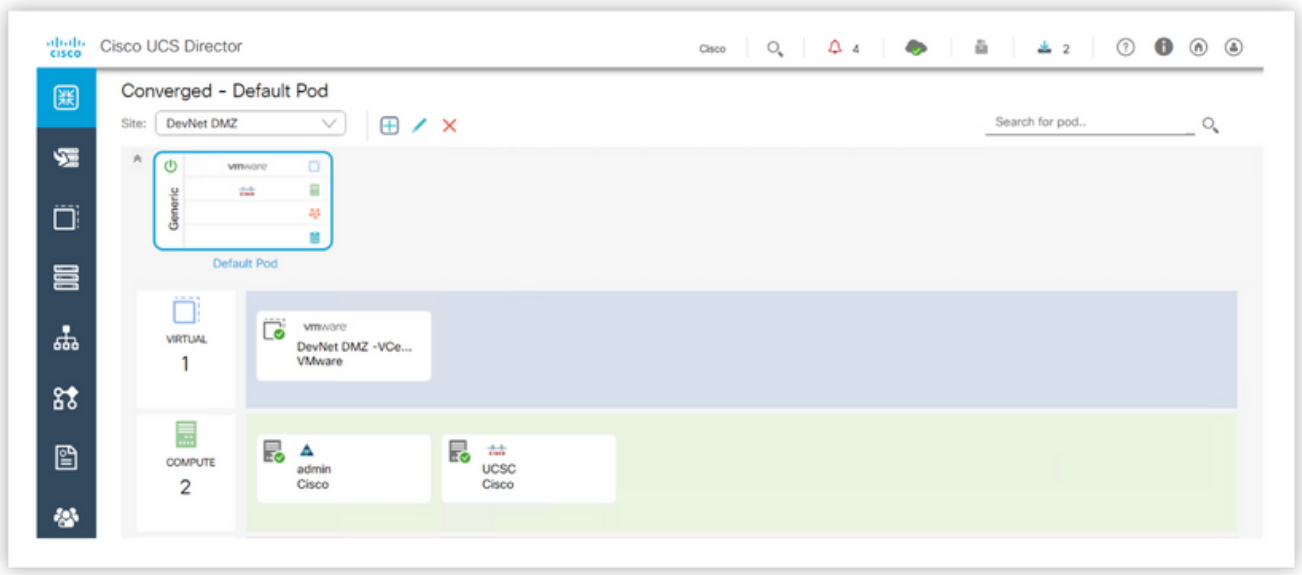
After they are built and validated, these workflows perform the same way every time, no matter who runs the workflows. An experienced data center administrator can run them, or you can implement role-based access control to allow your users and customers to run the workflows on a self-service basis, as needed.

With Cisco UCS Director, you can automate a wide array of tasks and use cases across a variety of supported Cisco and non-Cisco hardware and software data center components. A few examples of the use cases that you can automate include, but are not limited to:

- VM provisioning and lifecycle management
- Network resource configuration and lifecycle management
- Storage resource configuration and lifecycle management
- Tenant onboarding and infrastructure configuration
- Application infrastructure provisioning
- Self-service catalogs and VM provisioning

- Bare metal server provisioning, including installation of an operating system

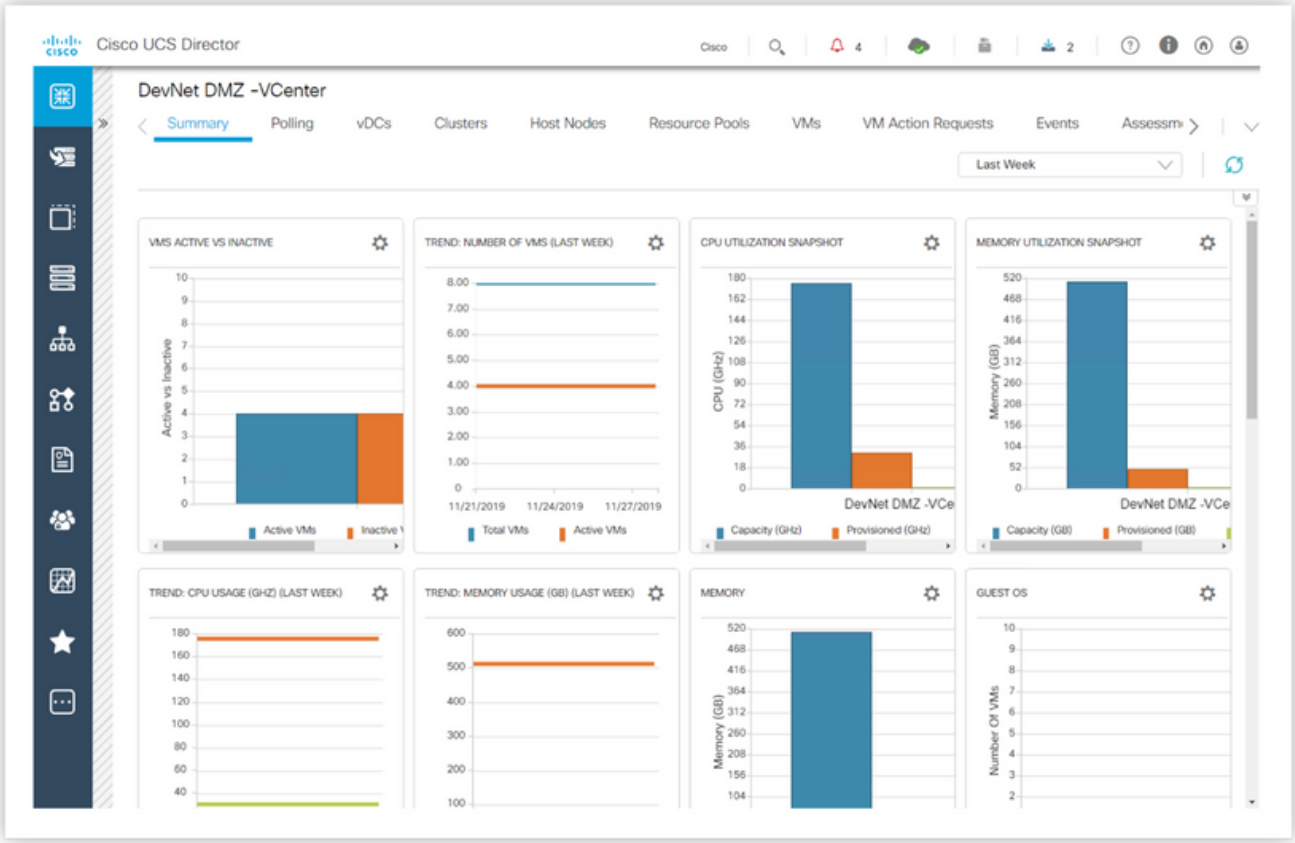
UCS Director Converged



Features and benefits

- **Central Management** – Provides a single interface for administrators to provision, monitor, and manage the system across physical, virtual, and bare metal environments. Unified dashboards, reports, and heat maps, which reduce troubleshooting and performance bottlenecks.
- **Self-service Catalog** – Enables end users to order and deploy new infrastructure instances conforming to IT-prescribed policies and governance.
- **Adaptive Provisioning** – Provides a real-time available capability, internal policies, and application workload requirements to optimize the availability of your resources.
- **Dynamic Capacity Management** – Provides continuous monitoring of infrastructure resources to improve capacity planning, utilization, and management, and identifies underutilized and overutilized resources.
- **Multiple Hypervisor Support** – Supports VMware ESX, ESXi, Microsoft Hyper-V, and Red Hat hypervisors.
- **Computing Management** – Provisions, monitors, and manages physical, virtual, and bare metal servers, as well as blades. Enables end users to implement virtual machine life-cycle management and business continuance through snapshots. Enables administrators to access server utilization trend analysis.
- **Network Management** – Provides policy-based provisioning of physical and virtual switches and dynamic network topologies. Enables administrators to configure VLANs, virtual Network Interface Cards (vNICs), port groups and port profiles, IP and Dynamic Host Control Protocol (DHCP) allocation, and Access Control Lists (ACLs) across network devices.
- **Storage Management** – Provides policy based provisioning and management of filers, virtual Filers (vFilers), Logical Unit Numbers (LUNs), and volumes. Provides unified dashboards that enable administrators comprehensive visibility into organizational usage, trends, and capacity analysis details.
- **Dashboards** – Provides multiple dashboards to track resource and policy utilization.

UCS Director Dashboard



Model-based orchestration

Cisco UCS Director includes a task library that contains over 1000 tasks and out-of-the-box workflows. Model-based orchestration and a workflow designer enable you to customize and automate the infrastructure administrative and operational tasks. You can extend and customize the system to meet individual needs.

Cisco UCS Director programmability

Cisco UCS Director programmability is flexible and provides a variety of methodologies to automate. UCS Director is an automation and orchestration platform with several programmatic capabilities.

- **REST API** - The REST API and the REST API Browser enable the management of UCS Director. UCS Director handles the management of policies, groups, virtual entities, etc. The REST API also provides a way to launch UCS Director workflows.
- **Tasks** - Tasks are either provided out-of-the-box or can be created as custom tasks. Tasks can be atomic or monolithic and are written in CloupiaScript, a JavaScript-like language.
- **Script Libraries** - Can contain CloupiaScript methods, Java jar files, lists of values, and more.
- **PowerShell Host** - Use PowerShell Cmdlets with UCS Director via the PowerShell Host.
- **Integrations** - UCS Director supports integrations with a variety of services to provide everything from issue tracking, to payment services, to source code control.

8.5.8

Cisco UCS Director – REST API

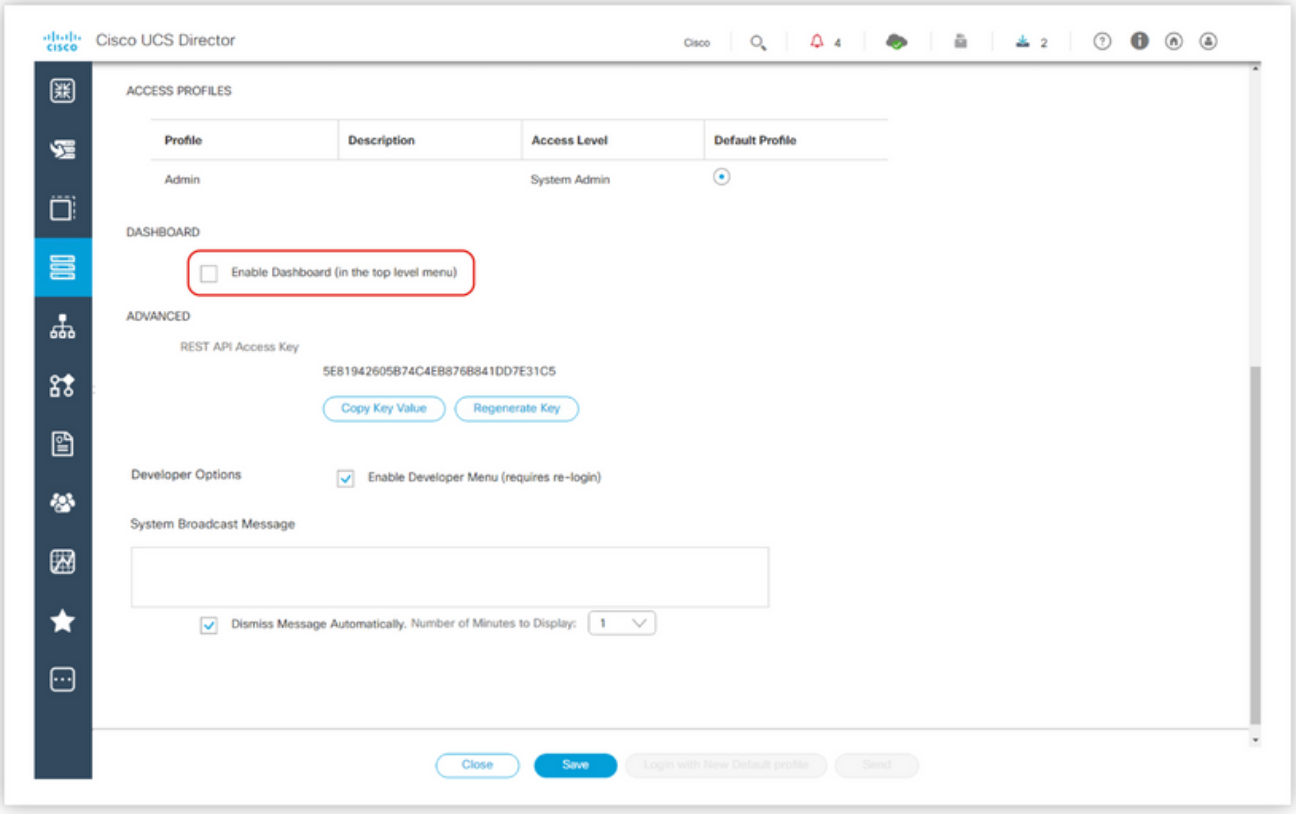
To access the REST API through Cisco UCS Director, a valid Cisco UCS Director user account and an API access key are needed. Cisco UCS Director uses the API access key to authenticate API requests. This access key is a unique security access key code that is associated with a specific Cisco UCS Director user account.

You must pass the REST API access key as an HTTP header in the following format:

```
X-Cloupia-Request-Key: F90ZZF12345678ZZ90Z12ZZ3456FZ789
```

Using the GUI

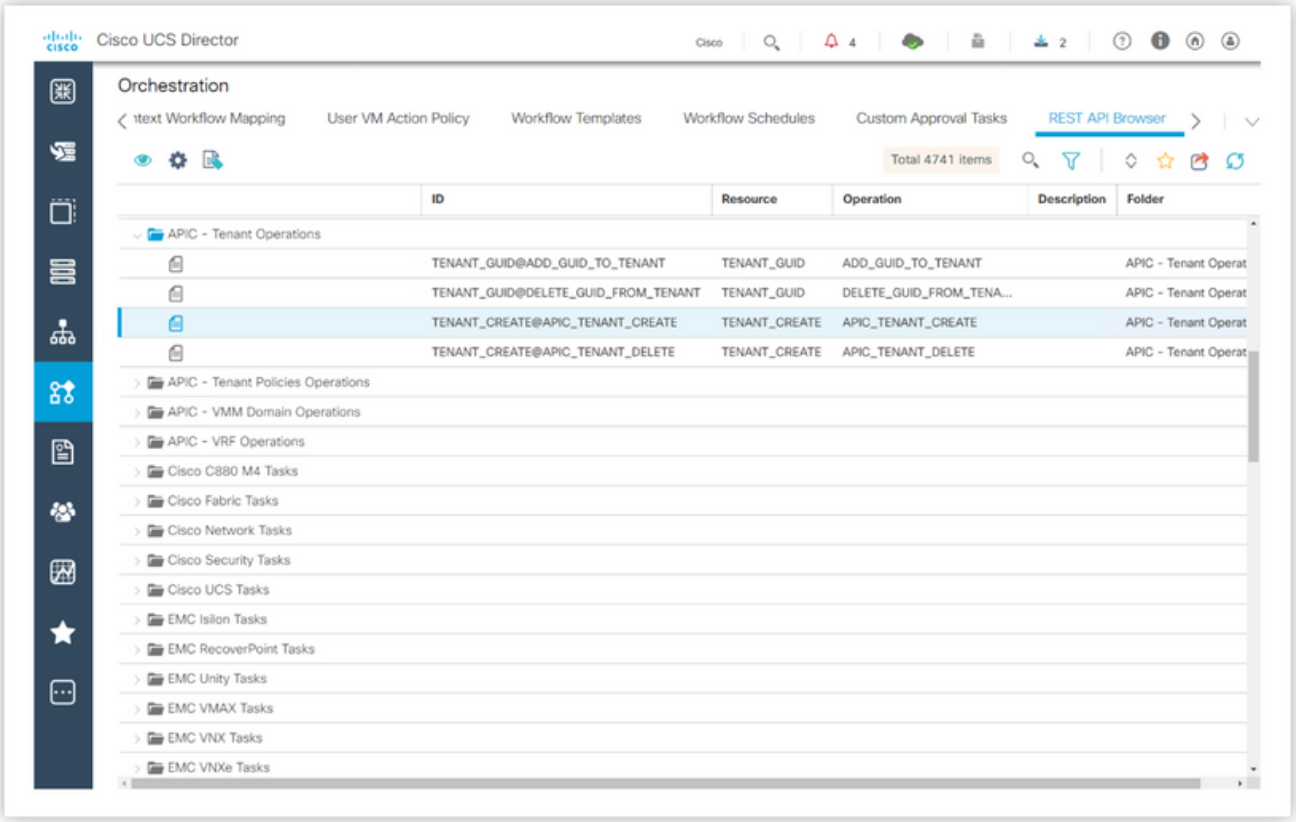
UCS Director Enable REST API Browser



When you enable the developer menu, Cisco UCS Director GUI provides a developer menu option for developers to access the report metadata and REST API Browser. You can then access the following features:

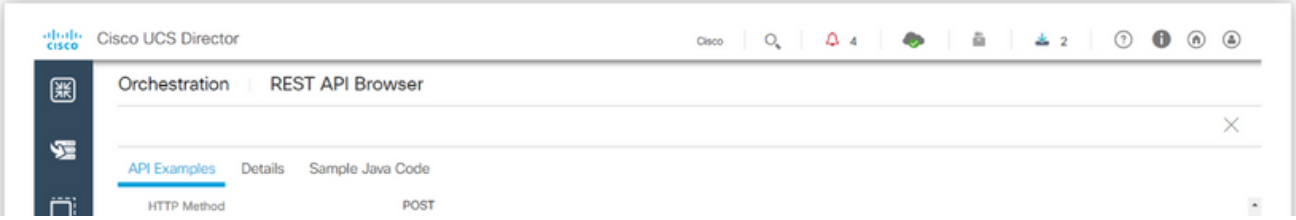
- **Report Metadata** – The report metadata enables you to view the REST API URL for every report displayed in Cisco UCS Director.
- **REST API Browser** – The REST API Browser is accessible from the Orchestration menu of Cisco UCS Director. The REST API Browser provides API information and API code generation capabilities that make it easy to see and work with all the available APIs, including both the REST APIs and the Java APIs.

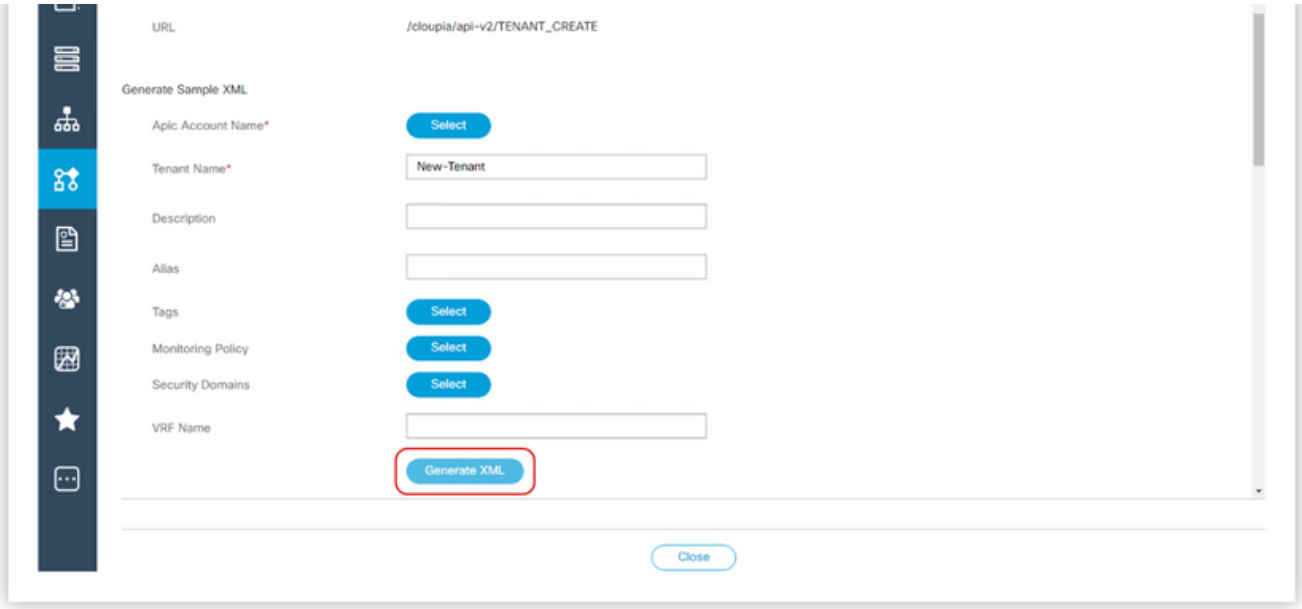
UCS Director REST API Browser



- **REST Client** – The REST Client is a useful widget for parsing and viewing API requests and responses. In this widget, you can enter a REST URL and apply an HTTP method such as POST, PUT, or DELETE to the URL for data manipulation. The REST Client provides a simple user interface for entering a URL to fetch data from the Cisco UCS Director server.

UCS Director REST Client





How to make a REST API request

API clients use an HTTP request to interact with Cisco UCS Director. To pass the REST API access key, each request must be associated with an HTTP header called X-Cloupia-Request-Key with its value set to the current REST API access key.

Requests made to the API have the following characteristics:

- They are sent over HTTP.
- Request format encoding can be either JSON or XML in UCS Director API Version 1.
- The request must contain a valid URL.

API VERSION 1

http://SERVER/app/api/rest?formatType=json&opName=operationName&opData=operationData

Where:

- `SERVER` – This is the IP address or the hostname of Cisco UCS Director.
- formatType – This is either JSON or XML. JSON is the default.
- `opName` – The API operation name associated with the request. For example,: `userAPIGetMyLoginProfile` or `userAPIGetVMActionStatus`.
- `opData` – Parameters (or arguments) associated with the operation.

Cisco UCS Director uses JSON or XML encoding of the parameters. If no arguments are required for the operation, use `{}` as an empty set. Before you send data in a request, encode the URL by applying escape characters as appropriate. For details about encoding the URL, see the RFC at http://www.ietf.org/rfc/rfc1738.txt.

API Version 2

Only XML is supported for Version 2 of the UCS Director API.

http://server/cloupia/api-v2/group

- HTTP method: POST.

```
<cuicOperationRequest>
  <payload>
    <![CDATA[
      <AddGroupConfig>
      <groupName>TestGroup</groupName>
      <groupDescription></groupDescription>
      <parentGroup>0</parentGroup>
      <groupCode></groupCode>
      <groupContact>jbesai@cisco.com</groupContact>
      <firstName></firstName>
      <lastName></lastName>
      <phone></phone>
      <address></address>
      <groupSharePolicyId></groupSharePolicyId>
      <allowPrivateUsers>false</allowPrivateUsers>
      </AddGroupConfig>
    ]]>
  </payload>
</cuicOperationRequest>
```

About operations data parameters or arguments

As the method and the API resource type are communicated through the `opName`, the operation parameters must present any arguments that you want to designate a specific instance of the resource to be operated upon.

Operations data parameter syntax

The following are the examples of operations data parameter syntax in JSON format.

- No parameters - `{}`
- One parameter; integer (for example, 10) - `{param0:10}`
- One parameter: string (for example, cloud) - `{param0:"cloud"}`
- Two parameters: a string and an Integer - `{param0:"cloud",param1:10}`
- Two parameters: a string with null value and an Integer - `{param0:null,param1:10}`
- Three parameters - {param0:"cloud",param1:"cloupia",param2:100}

Operation Data Parameter examples

```
…&opData={param0:"Create NFS Datastore",param1:{"list":[{"name":"Volume
Size","value":100},
{"name":"Select Group","value":"14"},{"name":"Select vDC","value":18}]},param2:212}
```

- `param0` – Name of the workflow being invoked through the REST API.
- `param1` – Input being passed to the workflow. If there is more than one input, separate the inputs with commas and put two single quotation marks around the input names and values. If there are no inputs, use the keyword null as the parameter value.
- `param2` – If this workflow is being invoked as a child workflow of another service request, use the service request (SR) ID. If this workflow is not invoked as a child workflow, use −1. When −1 is used, a new service request is created.

How to use cURL commands

cURL is a command line tool for getting or sending data using URL syntax. You can use the cURL command to execute a REST API request.

The following sample shows how to execute the `userAPISubmitWorkflowServiceRequest` API and pass the input values:

```
curl  -v -X POST -H 'X-Cloupia-Request-Key:5CF4C115F0034B189616B2B8EBA0F220' -g
'http://172.17.32.75/app/api/rest?
formatType=json&opName=userAPISubmitWorkflowServiceRequest&opData=
{param0:"TestWorkFlowFromAPI",param1:{"list":[{"name":"A1","value":"Hello"},
{"name":"A2","value":"World"}]},param2:-1}'
```

Custom tasks and CloupiaScript

Cisco UCS Director provides automated, profile-based provisioning, management, and reporting of infrastructure resources. Cisco UCS Director incorporates a powerful orchestration engine that enables complex operations on any element of your converged infrastructure, both physical and virtual. These operations are embodied in workflows, which are scripted sequences of individual tasks. Cisco UCS Director comes complete with a large library of tasks.

The tasks in Cisco UCS Director are written in CloupiaScript, a version of JavaScript with libraries that enable orchestration operations. With CloupiaScript, it is possible to embed scripts in workflow tasks and to write custom tasks.

SSH to a network device CloupiaScript example

```
function testSSHClient() {
        var client = new SSHClient(input.ipAddress, 22, // 22 = SSH standard port
input.userName, input.password);
        client.connect();
        var session = client.openShell(511,25); // (511, 25) = (columns, rows)
        var shellStream = new PrintStream(session.getOutputStream());
        shellStream.println(input.command);
```

```
        shellStream.flush();
        client.disconnect();
}
testSSHClient();
```

Additional UCS Director resources

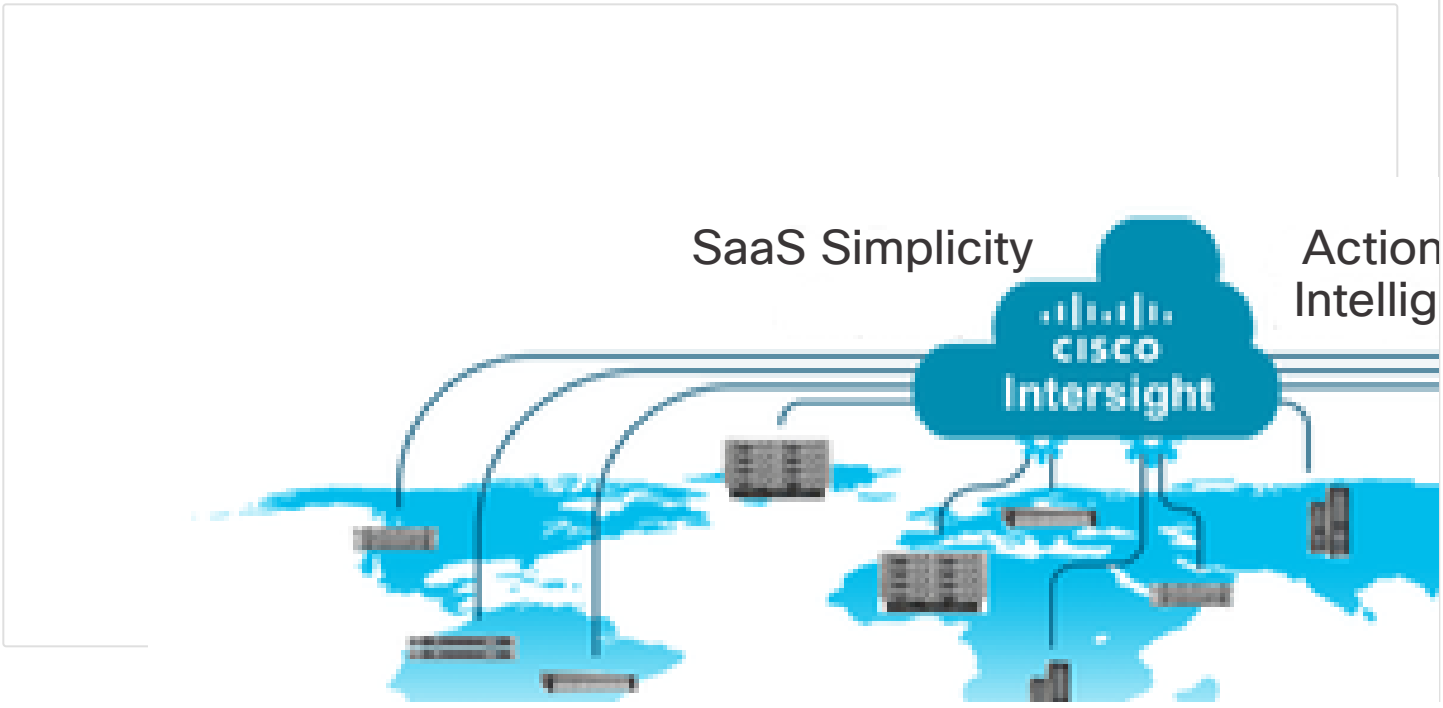
UCS Director REST APIs can be called from any programming language or tool that support HTTP requests. For example PowerShell and Python or Postman and cURL. CloupiaScript is used directly or imported into built-in and custom tasks. Tasks can also call PowerShell Cmdlets through the PowerShell agent. Built-in tasks and custom tasks are used to build workflows and workflows can be invoked from the UCS Director REST API, making UCS Director almost completely automatable

8.5.9

Cisco Intersight

Cisco Intersight was introduced in 2018 by Cisco Systems. Cisco Intersight is a Software as a Service (SaaS) systems management platform capable of managing infrastructure at the edge and remote locations, as well as in the data center. When using the service, you can scale infrastructure up or down as needs increase or decrease. Because it provides a REST API to access the Management Information Model (MIM), you can manage the infrastructure as code. The Intersight API is consistently available with a cloud-based management model. New features can be added to the service without impacting existing automation systems.

Cisco Intersight SaaS-based Management



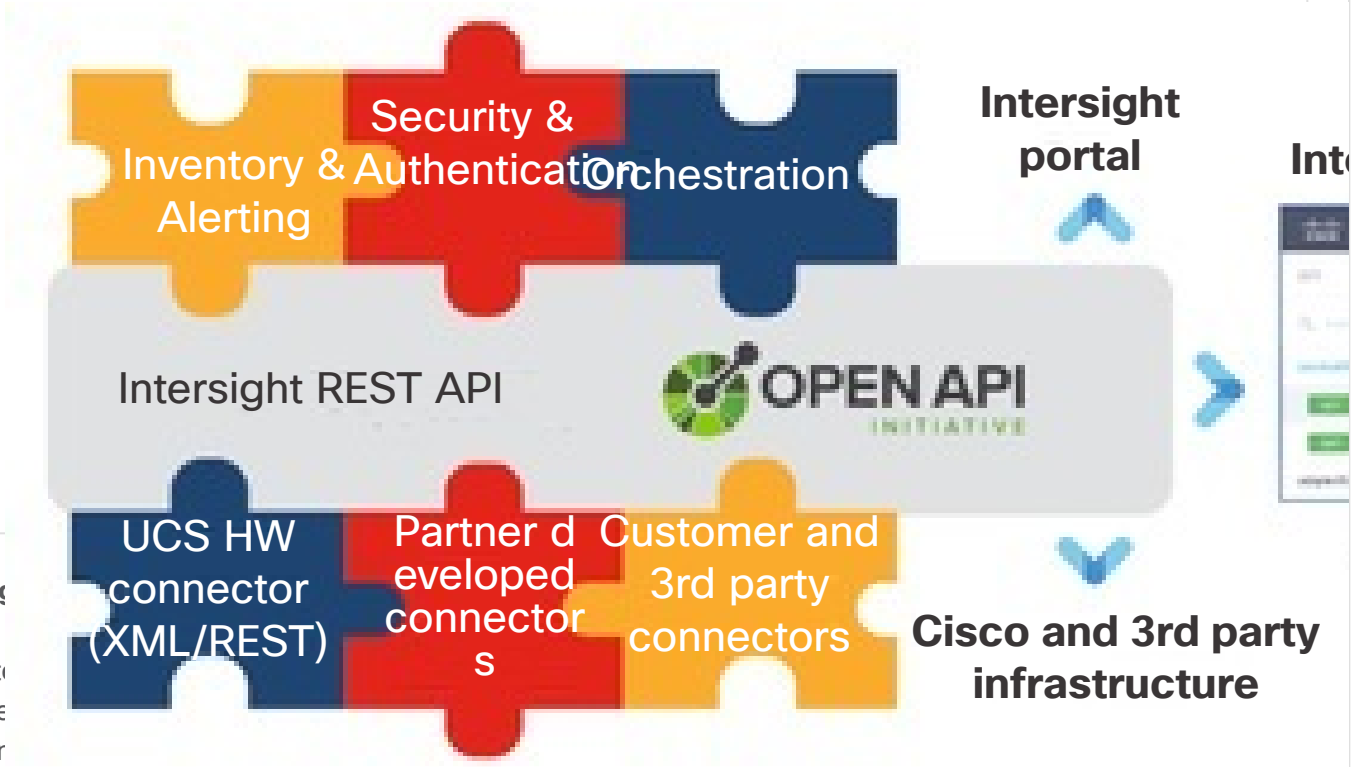
Intersight a

Cisco UCS
administrator. This abstraction layer means that an administrator has a standardized method for managing server components. The abstraction layer is presented as a web-based graphical user interface (GUI) and an application programming interface (API).

Cisco Intersight builds on the UCS Unified Fabric and UCS Management experience with a Cisco-hosted and maintained management platform. With a Server Profile, which performs model-based service provisioning in UCS, you can use Intersight to configure servers, manage resources, and ensure that policies align. With this approach, you avoid failures caused by inconsistent configuration.

With the Intersight API you can build integrations with Cisco Intersight for additional tasks like monitoring, analysis, configuration, deployment, and orchestration.

Cisco Intersight API Framework



Intersi

The Int
that are
the trar

All the physical and logical components visible in Cisco Intersight are represented in a hierarchical MIM, also referred to as the Management Information Tree or MIT. The MIT is a tree structure with nodes. Each node in the tree represents a managed object (MO) or group of objects that contains the nodes' administrative state and its operational state.

For more information, see the Management Information Model section of the Intersight documentation.

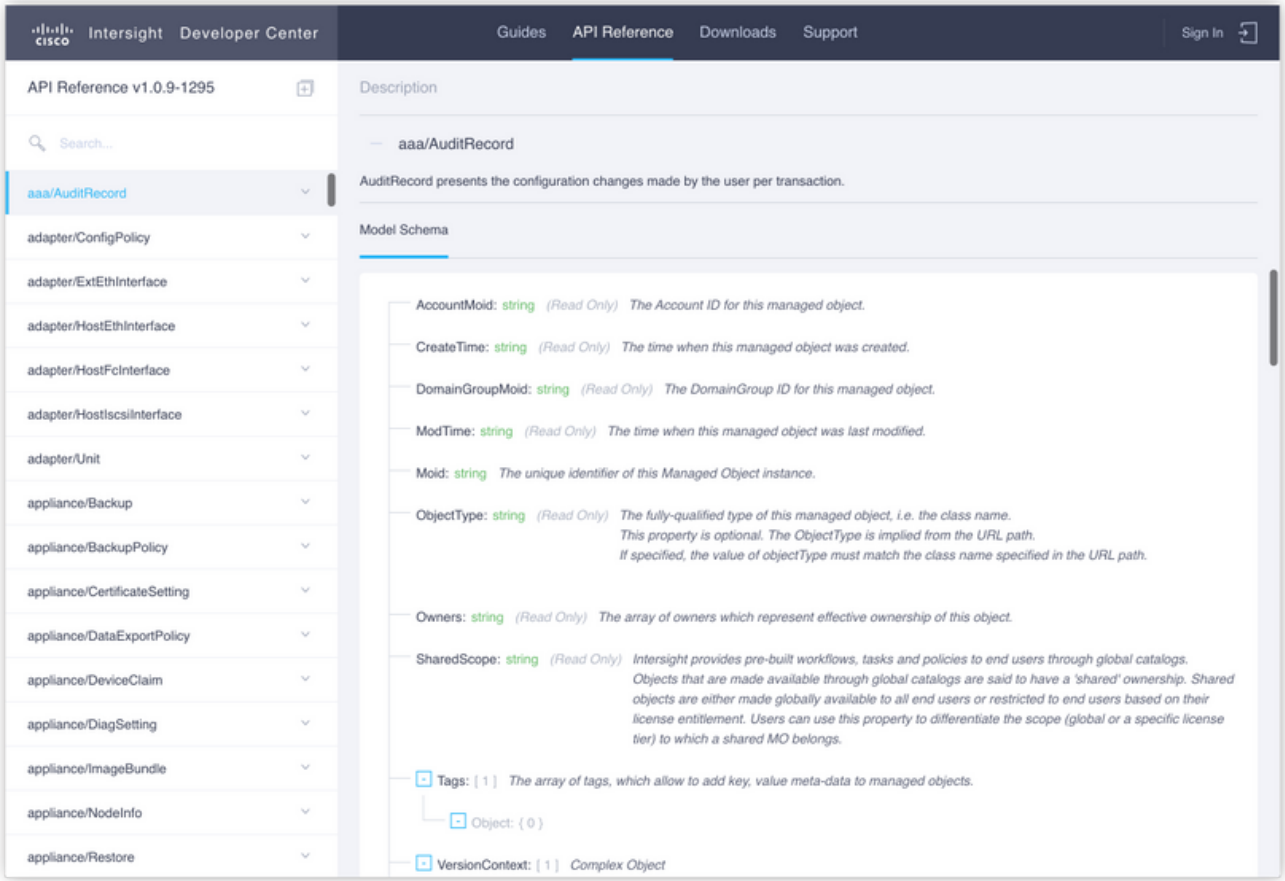
Accessing the Intersight API

There are a couple of ways to gain access to the Intersight API You can use a web browser as an Intersight API REST Client, or API keys for remote or service access.

Intersight API REST client

To view and interact with the Intersight API, access https://intersight.com/apidocs in a browser. The **API Reference** pages hosted on intersight.com contain live information on the complete Intersight resource model. Resources can be searched by name and the **Model Schema** can be viewed.

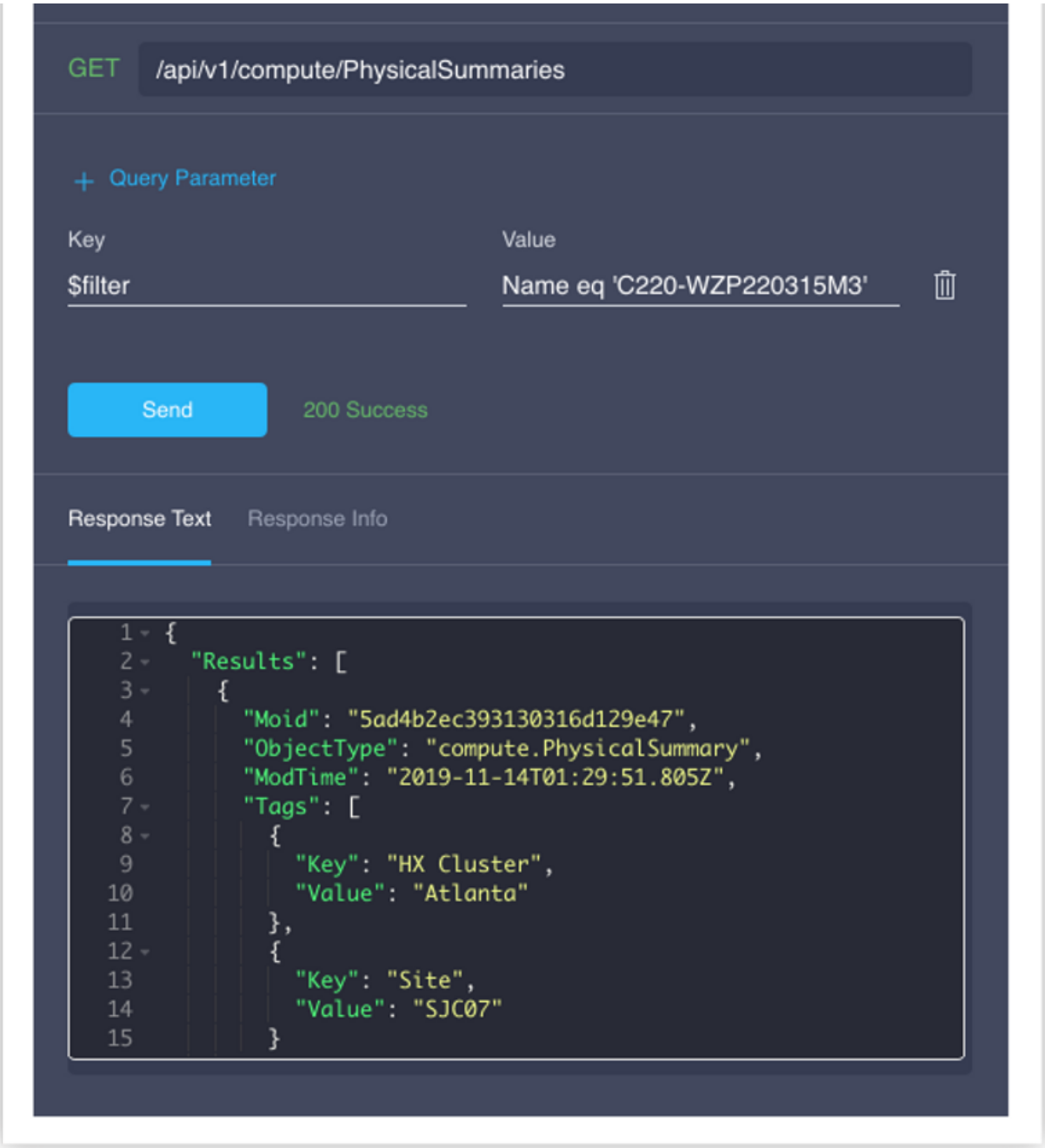
Intersight API Reference and Model Browser



The site at intersight.com also hosts a **REST Client** that allows direct interaction with the API.

The **REST Client** on the site also supports the full query language of the API and details on the supported **Query Parameters** can be viewed in the Parameters pages.

Intersight API REST Client



Intersight API keys

You create API keys for your Intersight account for typical remote access through SDKs or other programming environments. For example, API keys are used by Intersight Ansible modules and key information is specified in Ansible playbooks. The playbooks also contain variables that you can edit as needed.

Intersight API keys are categorized into two; an API key ID and a secret key. The API Key ID is a multi-character string that is always visible after initial key creation. The secret key is an RSA Private Key that is only available at API key creation. To create API keys in Intersight, you must login at https://intersight.com and perform the following steps:

Step 1. Click the Settings icon.

Step 2. Click **API Keys** in the left-hand navigation pane.

Step 3. Click **Generate API Key**.

Step 4. Enter a **Description** for the key.

Step 5. Click **Generate**.

Step 6. Click the **Save Secret Key to text file** icon. A "SecretKey.txt" file is downloaded to your default downloads location.

Note: You can create Intersight API keys only if you have the required product licensing.

Ansible Modules for Cisco Intersight

Intersight supports several API integrations, including Ansible modules that enable inventory collection and configuration management of Intersight resources. Ansible is written in Python, and the modules for Intersight interact with the API using the code written in Python. This securely authenticates with Intersight using API keys and standard Ansible modules for URL based access.

Several example playbooks and Lab guides are hosted on GitHub.

Additional SDKs and resources

In addition to Ansible Modules, Intersight provides a Python SDK and PowerShell modules that are generated from Intersight OpenAPI schema (which is also referred to as the Intersight Swagger Spec). The intersight.com site hosts the latest OpenAPI specification and links to the SDKs.

8.4
Cisco Network Management

8.6
Cisco Collaboration Platforms