

Controller Service Descriptions

Cisco APIC Enterprise Module FCS - Release 1.0

February 2014



Table of Contents

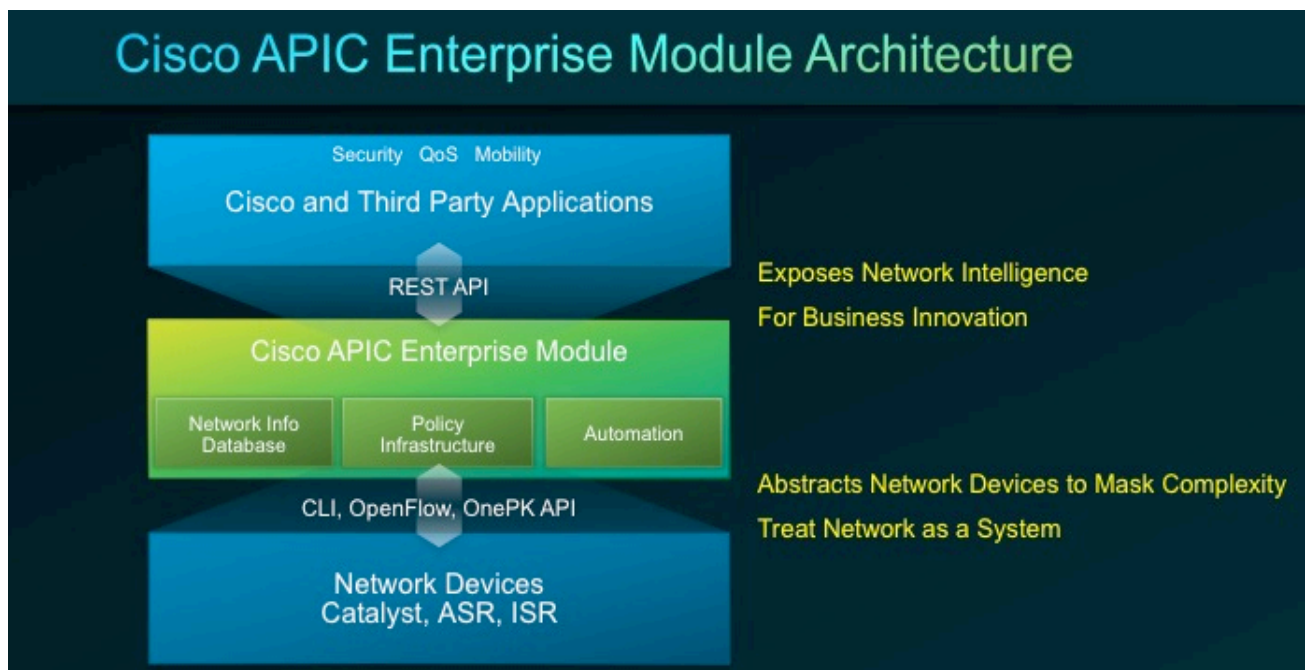
1.0 Introduction.....	3
2.0 Service Description and Value Propositions.....	5
2.1 Role Based Administrative Controls.....	5
2.2 Inventory DB browser and API access	6
2.3 Network and User Visualizer.....	7
2.4 Identity Services	9
2.4.1 Identity Services (Native)	10
2.4.2 Identity Services - Enhanced	10
2.5 QoS Policy visibility and control	11
2.6 ACL Policy visibility and control	12
2.7 Zero Touch Deployment.....	13
2.8 Application Visibility (native).....	15
2.9 SDN Application enablement with REST APIs.....	16
Appendix.....	17
1.0 APIC Enterprise Module: North-Bound REST API.....	17

1.0 Introduction

The Cisco Application Policy Infrastructure Controller (APIC) Enterprise Module allows IT organizations to have complete visibility into their networks, automating network and policy configuration while managing applications across the WAN and access networks. From a single x86 or virtualization-enabled computer, you can quickly deploy branch and campus network services, policies, and applications.

A key component of the Cisco Open Network Environment (ONE) Enterprise Networks Architecture, the Cisco APIC Enterprise Module can be used across wired, wireless, physical, and virtual networks. It protects your investment by working with your existing infrastructure. And it lets you create an intelligent, open, programmable network that helps you do the following:

- Quickly respond to growing application needs
- Free up time to innovate for business opportunities
- Ease the complexity of mobility/bring your own device (BYOD), cloud, and other trends



The Cisco APIC Enterprise Module is constructed of three elements: a consolidated network information database, policy infrastructure and automation. Combined, that can substantially increase network automation and agility, significantly lowering the

time that IT spends on operational activities. It provides the **unique ability to see the entire network as a single entity**, instead of individual network elements. The result is reduced network complexity, accelerated application rollout across wired and wireless infrastructure, and more efficient network management and troubleshooting.

The Cisco APIC Enterprise Module can run on any x86 server with:

- vSphere 5.x Hypervisor
- 8-core vCPUs and 16 GB or more of RAM

Chrome is the only supported browser for the controller user access.

This controller supports the overall Cisco Enterprise Networking Group product portfolio, including Cisco Integrated Services Routers (ISRs), Cisco Aggregation Services Routers (ASRs), Cisco Catalyst switches, Wireless LAN Controllers and Access Points.

2.0 Service Description and Value Propositions

2.1 Role Based Administrative Controls

Service Description

The Cisco APIC Enterprise Module supports Role Based Administrative Control (RBAC) for restricting controller access to authorized users. RBAC on the controller will support pre-defined roles for administrative control as well as user-defined roles based on device tag domains. The following roles are pre-defined in the controller:

- Admin – complete administrative privileges and can assign users to other roles
- Operator - able to write to policy, QoS, topology etc. but not the setup API, tftp-server, RADIUS-proxy etc.
- Policy – able to write to policy API
- Observer – read only access

Apart from the above pre-defined roles, the controller would also support user-defined roles for domain control. The domains are defined via device tags. Within a domain, we can then assign users the above pre-defined roles – Admin, Operator, Policy, and Observer. A user can have different roles assigned in different domains.

Here are more details on the controller RBAC workflow -

- Upon initial installation of the controller, we'll have a default "Admin" user that the customer can use to log in and configure the product. This admin user would have a default password which the customer can change after logging in the first time.
- The "Admin" user would be able to add more admin users as well as non-admin users
- Since tags are associated to authorization, only "Admin" users would be able to define tag domains, and then assign non-admin users roles (operator, policy, observer) on those tag domains.
- Non-admin users would then be able to log in to manage their assigned domains

2.2 Inventory DB browser and API access

Service Description

The Cisco APIC Enterprise Module periodically scans the network to create a Network Information Database that serves as a “single source of truth” for the network administrator/IT. This inventory includes all network devices, along with an abstraction for the entire network provides full awareness and awareness of the overall operational health of the physical network. This inventory database allows the applications to be device agnostic, so that the configuration differences between devices is not an issue. This database also enables real-time network inventory and asset service management.

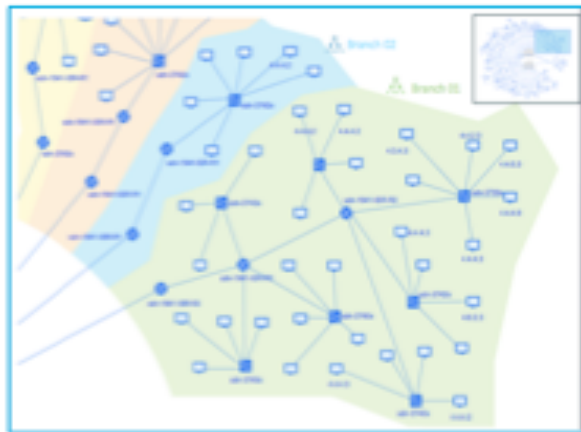
- Quick, easy and efficient network discovery using IP range or Cisco Discovery Protocol (CDP)
- The user can start, stop and delete the scan anytime
- The network inventory is presented in the form of a table and which includes information such as device name, device type, IP address, mac-address, SW version, connected hosts, device roles, reachability status etc.
- Network administrator is able to assign Geo Site (location) and Custom tags for complete flexibility in grouping and classification of devices based on business logic e.g. lines of business, service mix etc.
- Hosts attached to the network devices are discovered via CDP, LLDP and IP device tracking
- Detailed inventory information for easier consumption by various controller services and applications
- Real-time network inventory and asset service management
- Allows manual entry of network devices not supporting CDP
- The controller inventory database can scale up to 1000 network devices, excluding hosts

Device Status	Device Name	MAC Address	IP Address	Hosts	IOS/Firmware	Platform	Serial Number	Configuration	Device Role	Location	Tag	Le Uj Ti
Reachable	SDN-CAMPUS-C3850	18:9C:5D:DA:75:77	20.1.1.1	17.7.7.15	03.02.02.SE	WS-C3850-48P	FOC1743X0CZ	View	<input type="button" value="Distribution"/>	Add	Corp Finance	2017
Reachable	SDN-BRANCH-L-C3750E	00:1D:71:00:82:C0	4.4.4.2		15.0(1)SE3	WS-C3750E-24TD	CAT1133W0DY	View	<input type="button" value="Distribution"/>	Add	HR	2017
Reachable	SDN-CAMPUS-C6509E	00:18:74:9E:1C:00	14.4.4.2		12.2(33)SXJ6	WS-C6509-E	SMG1027NQN	View	<input type="button" value="Distribution"/>	Add	Corp Finance	2017
Reachable	SDN-BRANCH-L-C3750X	00:27:0D:3B:D1:C0	4.4.4.3	192.168.20.42, 192.168.20.41, 192.168.20.40	15.0(1)SE3	WS-C3750X-48P	FDO1348K00L	View	<input type="button" value="Access"/>	Add	HR	2017

2.3 Network and User Visualizer

Service Description

The Cisco APIC Enterprise module auto-discovers and maps network devices to a physical topology with detailed device level data. Its auto visualization feature enables you to have a highly interactive mechanism for viewing and troubleshooting your network. The controller topology view provides complete awareness and visibility of the overall operational health of the physical network and is a key enabler for end-to-end network visualization.



- Physical topology includes network devices, connecting links, interfaces and connected hosts
- The topology view will show wired hosts attached to the appropriate network device

- The topology view will show the wireless end points/hosts connected to the appropriate Access Point
- With appropriate Identity Services enabled, user names are stored and displayed along with their related hosts
- Ability to display the L2 or L3 (Static, OSPF, ISIS, PBR, ECMP) topology on top of the physical topology.
- Detailed information about every node and link is displayed. This information consists of device name, device role, IP address, interface name etc.
- Detailed information for each host can be displayed. This information includes userid, wired/wireless and location data.
- User can view a topology with groups. The grouping information comes from the inventory database
- User can zoom in a selected region by selecting the region in the topology GUI.

2.4 Identity Services

Service Description

The Cisco APIC Enterprise module is able to gather user identity information via RADIUS proxy, via Active Directory through LDAP calls or via the Cisco Identity Services Engine (ISE). This is an important component in building secure network access service across wired, wireless and VPN networks. The Identity information collected is a key enabler for building highly sophisticated policies with user level information for tighter enforcement.

Identity Services	APIC-EM Capabilities
No Integration	<ul style="list-style-type: none">• Identity based policy enablement and reporting cannot be done using the Controller
RADIUS	<ul style="list-style-type: none">• Controller can act as RADIUS Proxy• Collects Identity information from the RADIUS authentication exchange between AAA server and end host (User Id, IP address, mac-address)• Network device configuration change needed to point to the Controller (instead of AAA server)• 802.1x configuration needs to be enabled on the network
Active Directory	<ul style="list-style-type: none">• User group information from AD server via LDAP calls (used for enabling user group based policies)• Identity support through AD is a road map item, not available in APIC-EM Release 1.0
Identity Services Engine (ISE)	<ul style="list-style-type: none">• pxGrid connection agent support for communication with Cisco ISE• Cisco ISE provides identity and device type information for each user• No configuration change needed on network device which is already using ISE• 802.1x configuration needs to be enabled on the network

2.4.1 Identity Services (Native)

The Cisco APIC Enterprise module can be configured as a RADIUS Proxy to collect user identity information or can gather user group information from Active Directory through the LDAP calls. This identity data is available to other controller services such as inventory, topology, policy-manager etc.

- Real-time end point visibility and user identity tracking (userid, IP, End Point connectivity)
- Support for RADIUS proxy and LDAP integration with third party identity management solutions such as FreeRADIUS, Active Directory etc.
- Collects Identity information (userid, IP address, mac-address) from the RADIUS authentication exchange between AAA server and client (end host)
- Ability to get user group information from AD through LDAP calls (used for enabling user group based policies)

2.4.2 Identity Services - Enhanced

The Cisco Identity Service Engine (ISE) is a security policy management and control platform. It automates and simplifies access control and security compliance for wired, wireless and VPN connectivity. Cisco ISE is primarily used to provide secure access and guest access, support BYOD initiatives, and enforce usage policies. The Cisco Platform Exchange Grid (pxGrid) controller function is supported on Cisco ISE.

The Cisco APIC EM controller supports the pxGrid connection agent function and can communicate with ISE to obtain user and device information. This enables the controller to do highly sophisticated per-user policy enforcement, which can improve mobile security policies (including BYOD) across the enterprise/campus network.

- Supports pxGrid connection agent for contextual information exchange with the Cisco Identity Services Engine (ISE)
- The controller has the ability to get real-time user identity (userid, IP) and device type from ISE
- This Identity data is available to other controller services, such as the policy manager for per-user policy enforcement.

2.5 QoS Policy visibility and control

Service Description

The QoS application in the Cisco APIC Enterprise Module enables the network administrator to automate QoS policy deployment across the entire network, or to just a subset of network elements (e.g., to a domain per device tags). QoS deployment and change management across the entire enterprise network is facilitated through an advanced GUI. The network admin can quickly set and enforce QoS policies across the network with a single click of the button ensuring that the network devices will remain in compliance. This helps keep application traffic behaving consistently and in accordance with the QoS service-level agreements (SLAs).

QoS policy deployment –

- Uses Cisco Validated Design (CVD) classification and marking schema to assign well known and statically defined applications to desired class of service
- Allows easy customization of policies based on current needs and industry standards
- Supports one click QoS policy deployment and removal across the entire network or a subset of the network defined by tags
- The controller is able to automatically translate business level requirements into network level configuration to enable QoS across heterogeneous environment for quick implementation and leverage device capabilities to its full potential
- Supports policy enforcement for both wired and wireless hosts
- Supports time-based policies and delta policy updates
- Modification of already existing policy via the APIC Enterprise Module is supported

2.6 ACL Policy visibility and control

Service Description

The Cisco APIC Enterprise Module provides consistent methodology across applications for analytics gathering, situation analysis, and policy decision making which can be enforced manually or programmatically. It queries and analyzes ACLs on each network devices. This helps the network admin to simplify and speed troubleshooting issues by quickly identifying ACL misconfigurations. The controller is able to auto-translate business policy into network device-level policy for quick and enforcement of policies across heterogeneous network device environment. It gathers analytics that help IT initiate policy enforcement across the enterprise network.

Policy Definition

- User can mark specific application traffic from a specific user with an appropriate DSCP value to influence how this application traffic is treated inside the network
- User can define a policy using userid and statically defined applications with permit, deny and copy actions
- User can use Priority Level and Destination as action properties for the policies
- Flexible scheduling with user defined timers such as start/end time, idle/hard timeout, recurrence etc.
- Policies once deployed are removable with one touch operation

ACL Analysis

- Enables inspection, interrogation and analysis of network access control policies
- Fast and easy comparison of ACLs between devices to visualize differences and identify misconfigurations
- Ability to trace application specific paths between end devices to quickly identify ACLs in use and problem areas
- Enables ACL change management including easy identification of conflicts and shadows in the TCAM

2.7 Zero Touch Deployment

Service Description

Zero touch deployment aims to provide enhanced, secure and integrated solution for enterprise network customers to ease new network device rollout process or provisioning updates to an existing network. When the controller's scanner discovers a network device, it creates a database entry for this device that allows the network admin to provision the device with appropriate image and configuration. This capability eliminates manual intervention, saving time and potential errors.

Use Cases

1. New device provisioning (Ad-hoc OR Un-Claimed device provisioning)

- This use case is for the scenario where the new device is not part of a site roll out process or the admin has not pre-provisioned it in the Controller
- New device boots up and is able to discover the Controller and show up automatically in the Inventory database
- Network admin can then provision this device by specifying the configuration file and image information

2. RMA

- Network admin prepares the Controller for possible upcoming RMA by identifying device from Controller inventory as possible RMA candidate
- Installer plugs-in new device replacing existing mal-functioning device (like to like: same device type and model)
- New device boots up and is able to discover the Controller and show up automatically in the Inventory database
- The Controller does a match of this new device against the RMA device list to see find out exact device match (match based on device type, neighbor info etc.)
- Newly plugged-in device get the same configuration and image as the old device

3. New Site Provisioning

- Network admin provisions the entire site / branch via this use case (new site or clone an existing site)

- All Cisco devices for this new site is pre-provisioned in the Controller inventory (Serial number, model info etc.) with image and configuration file information
- As and when the new devices are powered up, the Controller is able to push the image and configuration to this new device

Platform support

Routers Platforms (IOS PI 24, IOS-XE 3.12) -

- Cisco ISR G2 (800, 1900, 2900, 3900)
- Cisco ASR 1000
- Cisco ISR G3

Switch Platforms (IOS-XE 3.6)

- Cisco Catalyst 4500
- Cisco Catalyst 4900
- Cisco Catalyst 3K (3560-C, 3560-X, 3750-X, 3650, 3850)
- Cisco Catalyst 2K (2960-C, 2960-XR, 2960-X, 2960-SF, 2960-S)

2.8 Application Visibility (native)

Service Description

The Cisco APIC Enterprise Module gives the network admin the ability to pull real time information from the network on a per application basis. The controller is able to provision the network devices for application visibility and thus eliminating the need for manually configuring the nodes for application visibility keeping in mind the different capabilities of the different network devices. This data can be used to rollout network wide QoS policies, react to network conditions (example: link saturation) by prioritizing certain applications over others etc.

Once the network admin has selected to deploy Application visibility on the enterprise network (or a subset of the network nodes), the controller will configure the network for application monitoring (NBAR2, Flexible Netflow). This enables the controller to collect data regarding the various applications running on each network device in the enterprise network.

The network visualizer view will have continuous updates about –

- Applications detected at various sites
- Percent of traffic covered by fully identified Applications
- Suggestions for custom App definitions with reasoning based on recurring IPs, DSCP values etc. of unknown application traffic flow

2.9 SDN Application enablement with REST APIs

Service Description

The rich set of open, North-Bound REST APIs supported by the Cisco APIC Enterprise Module enables 3rd party partners to develop applications for enhanced visibility into the IT networks, automating network and policy configuration while managing applications across the WAN and access networks. It enables the policies to automatically adapt to network changes that would be very difficult to configure otherwise. These APIs are available via the Cisco Developer Program ([DevNet](#)) to all Cisco developers and other third-party application developers.

One such example of REST API based application enablement is to address security concerns. The controller is able to automate network-wide rapid threat detection and mitigation by integrating and automating the Cisco Sourcefire security solution. When the Sourcefire application detects a security threat, it sends a request to the APIC Enterprise Module controller via the REST APIs to block the host causing this security threat. The controller is then able to find out the network node and the port to which this host is connected using the network inventory database and take appropriate remedial action.

Some of the partners who have already developed applications utilizing these APIs are Citrix, Glue Networks, Radware and Action Packed Networks etc.

Appendix

1.0 APIC Enterprise Module: North-Bound REST API

discovery

HTTP Method	URI	Description
POST	/discovery/	Starts a new discovery process and returns a discovery-id
GET	/discovery/{discovery-id}	Returns the properties of the discovery-id discovery-job
GET	/discovery/{discovery-id}/network-device/{start}/{end}	Returns the network devices discovered in the given range, start & end are 32 bit numbers, start < end
GET	/discovery/{discovery-id}/network-device/	Returns all the network-devices discovered in discovery-id discovery-job
GET	/discovery/count	Returns the number of discovery jobs
GET	/discovery/{start}/{end}	Returns the properties of discovery jobs in the given range, start & end are 32 bit numbers, start < end
GET	/discovery/	Returns the properties of all discovery jobs
PUT	/discovery/	Updates an existing discovery specified by discovery-id - only for starting/stopping the discovery the discovery-id is specified in the request object. If the request object says ALL, then all the discovery jobs are updated
DELETE	/discovery/{discovery-id}	Deletes the discovery specified by discovery-id
DELETE	/discovery/{start}/{end}	Deletes the properties of discovery jobs in the given range, start & end are 32 bit numbers, start < end
DELETE	/discovery/	Deletes the properties of all discovery jobs

discovery-frequency

HTTP Method	URI	Description
POST	/discovery-frequency/network-device/	Inputs network-device discovery frequency
POST	/discovery-frequency/interface/	Inputs interface discovery frequency
POST	/discovery-frequency/host/	Inputs host discovery frequency
GET	/discovery-frequency/network-device/	Returns network-device discovery frequency
GET	/discovery-frequency/interface/	Returns interface discovery frequency
GET	/discovery-frequency/host/	Returns host discovery frequency
GET	/discovery-frequency/	Returns all frequencies
DELETE	/discovery-frequency/network-device/	deletes network-device discovery frequency
DELETE	/discovery-frequency/interface/	deletes interface discovery frequency
DELETE	/discovery-frequency/host/	deletes host discovery frequency
DELETE	/discovery-frequency/	deletes all frequencies

device-credential

HTTP Method	URI	Description
POST	/device-credential	Inserts the device credential
GET	/device-credential	Gets all available device credentials
DELETE	/device-	Deletes the specified credential set

	credential/username/{username}/password/{password}	
--	--	--

network-device

HTTP Method	URI	Description
GET	/network-device/{network-device-id}	Returns complete information of a network-device
GET	/network-device/{network-device-id}/brief	Returns brief information on a network-device
GET	/network-device/count	Returns the number of network devices in the controller
GET	/network-device/{start}/{end}	Returns complete information of the network-devices in the given range, start & end are 32 bit numbers, start < end
GET	/network-device/	Returns complete information of all the network-devices stored in the controller

network-device-reachability-info

HTTP Method	URI	Description
GET	/network-device/{network-device-id}/reachability-info	Returns complete information of a network-device's reachability info
GET	/network-device/reachability-info/{start}/{end}	Returns complete information of all the network-devices' reachability info in the given range, start & end are 32 bit numbers, start < end
GET	/network-device/reachability-info/	Returns complete information of all the network-devices' reachability info

network-device-config

HTTP Method	URI	Description
GET	/network-device/{network-device-id}/config	Returns the network-device running-config
GET	/network-device/config/{start}/{end}	Returns the network-device running-config of all network-devices in the given range, start & end are 32 bit numbers, start < end
GET	/network-device/config	Returns the network-device running-config of all network-devices

network-device-role

HTTP Method	URI	Description
GET	/network-device/{network-device-id}/role	Returns the network-device role
GET	/network-device/role/{start}/{end}	Returns the network-device roles of all network-devices in the given range, start & end are 32 bit numbers, start < end
GET	/network-device/role	Returns the network-device roles of all network-devices
PUT	/network-device/role	Updates the network-device role (or roles of a list of network-

		devices) with the network-device-ids and role specified in the Request Object, if network_device-id="ALL", it is updated for all devices
--	--	--

role

HTTP Method	URI	Description
GET	/role/	Returns all the roles

role-network-device

HTTP Method	URI	Description
GET	/role/{role}/network-device/count	Returns all network-devices with given role
GET	/role/{role}/network-device/{start}/{end}	Returns all network-devices with given role in the number range, start & end are 32 bit numbers, start < end
GET	/role/{role}/network-device/	Returns all network-devices with given role

network-device-location

HTTP Method	URI	Description
POST	/network-device/location	Updates the network-device location (or locations of a network-device list) with the location specified in the Request Object The network_device-ids are specified in the request object. If the request object has "ALL" then all devices will be updated
GET	/network-device/{network-device-id}/location	Returns the network-device location
DELETE	/network-device/{network-device-id}/location	Deletes the network-device location

network-device-tag

HTTP Method	URI	Description
POST	/network-device/tag	Inserts a new tag for one or many network devices, the network-device-ids are specified in the request object
GET	/network-device/{network-device-id}/tag	Returns the network-device tag
GET	/network-device/tag/{start}/{end}	Returns the network-device tags of all network-devices in the given range, start & end are 32 bit numbers, start < end
GET	/network-device/tag/	Returns the network-device tags of all network-devices
DELETE	/network-device/{network-device-id}/tag/{tag}	Deletes the specified tag for the network device

line-card

HTTP Method	URI	Description
GET	/line-card/{line-card-id}	Returns line card information
GET	/line-card/{line-card-	Returns the network-device-id of the network-device

	<i>id</i> }/network-device	associated with the linecard
--	----------------------------	------------------------------

link

HTTP Method	URI	Description
GET	/link/{link-id}	Returns a link
GET	/link/network-device/{network-device-id}/count	Returns the number of links for the network_device-id
GET	/link/network-device/{network-device-id}/{start}/{end}	Returns all the links for the network device in the range, start & end are 32 bit numbers, start < end
GET	/link/network-device/{network-device-id}	Returns all the links for the network device
GET	/link/count	Returns the number of links in the controller
GET	/link/{start}/{end}	Returns all the links in the range, start & end are 32 bit numbers, start < end
GET	/link/	Returns all the links

link-tag

HTTP Method	URI	Description
POST	/link/tag	Inserts a new tag for one or many network devices, the link-ids are specified in the request object
GET	/link/{link-id}/tag	Returns the link tag
GET	/link/tag/{start}/{end}	Returns the link tags of all links in the given range, start & end are 32 bit numbers, start < end
GET	/link/tag/	Returns the link tags of all links
DELETE	/link/{link-id}/tag/{tag}	Deletes the specified tag for the link

interface

HTTP Method	URI	Description
GET	/interface/{interface-id}	Returns an interface
GET	/interface/network-device/{network-device-id}/count	Returns the number of interfaces for the network-device-id
GET	/interface/network-device/{network-device-id}/{start}/{end}	Returns all the interfaces for the network device in the range, start & end are 32 bit numbers, start < end
GET	/interface/network-device/{network-device-id}	Returns all the interfaces for the network device
GET	/interface/line-card/{line-card-id}/count	Returns the number of interfaces for the line-card-id
GET	/interface/line-card/{line-card-id}/{start}/{end}	Returns all the interfaces for the line-card in the range, start & end are 32 bit numbers, start < end
GET	/interface/line-card/{line-card-id}	Returns all the interfaces for the line card
GET	/interface/count	Returns the number of interfaces in the controller
GET	/interface/{start}/{end}	Returns all the interfaces in the range, start &

		end are 32 bit numbers, start < end
GET	/interface/	Returns all the interfaces

interface-tag

HTTP Method	URI	Description
POST	/interface/tag	Inserts a new tag for one or many network devices, the interface-ids are specified in the request object
GET	/interface/{interface-id}/tag	Returns the interface tag
GET	/interface/tag/{start}/{end}	Returns the interface tags of all interfaces in the given range, start & end are 32 bit numbers, start < end
GET	/interface/tag/	Returns the interface tags of all interfaces
DELETE	/interface/{interface-id}/tag/{tag}	Deletes the specified tag for the interface

interface-queue-statistics

HTTP Method	URI	Description
POST	/interface-queue-statistics/	Enable queue statistics collection for the given device IDs in the request If "device-id":ALL, enable for all active devices
GET	/interface-queue-statistics/{interface-id}	Get the interface statistics for the given interface-id
GET	/interface-queue-statistics/	Get the interface statistics for all the devices
DELETE	/interface-queue-statistics/{device-id}	Deletes the interfaces statistics collection for the given device-id
DELETE	/interface-queue-statistics/	Deletes the interface statistics collection for all devices

tag-network-device

HTTP Method	URI	Description
GET	/tag/{tag}/network-device	Returns all network-devices with given tag

tag-link

HTTP Method	URI	Description
GET	/tag/{tag}/link	Returns all links with given tag

tag-interface

HTTP Method	URI	Description
GET	/tag/{tag}/interface	Returns all interfaces with given tag

host

HTTP	URI	Description
------	-----	-------------

Method	URI	Description
GET	/host/{host-id}	Returns information about a host
GET	/host/count	Returns the number of hosts in the controller
GET	/host/{start}/{end}	Returns information about all the hosts stored in the range, start & end are 32 bit numbers, start < end
GET	/host/	Returns information about all the hosts stored in the controller
GET	/network-device/{network-device-id}/host	Returns all the hosts attached to the network-device

location

HTTP Method	URI	Description
POST	/location/	Creates a location
GET	/location/{location-id}	Returns a location by location-id
GET count	/location/count	Get the number of locations in the controller
GET	/location/{start}/{end}	Returns all the locations in the given range, start & end are 32 bit numbers, start < end
GET	/location/	Returns all the locations
PUT	/location/	Updates the location (or a list of locations) with the location-ids and attributes specified in the Request Object, if "location-id":"ALL", update all
DELETE	/location/{location-id}	Deletes a location by location-id
DELETE	/location/	Deletes all locations stored in the controller

location-network-device

HTTP Method	URI	Description
GET	/location/{location-id}/network-device/count	Returns the count of network devices at the location specified by {location-id}
GET	/location/{location-id}/network-device/{start}/{end}	Returns all network-devices belonging to location specified by {location-id} and in the range, start & end are 32 bit numbers, start < end
GET	/location/{location-id}/network-device/	Returns all network-devices belonging to location specified by {location-id}

external-aaa-keystore-file

HTTP Method	URI	Description
POST	/external-aaa-keystore-file/	Uploads a AAA server keystore file
GET	/external-aaa-keystore-file/	Returns list of uploaded AAA server keystore files

external-aaa-server-config

HTTP Method	URI	Description
POST	/external-aaa-server-config/	Creates or Updates aaa server config
GET	/external-aaa-server-config/	Returns aaa server config
DELETE	/external-aaa-server-config/	Deletes aaa server config

user

HTTP Method	URI	Description
GET	/user/{userid}	Returns information about a userid
GET	/user/count	Returns the number of users in the controller
GET	/user/{start}/{end}	Returns information about all the users in the given range, start & end are 32 bit numbers, start < end
GET	/user/	Returns information about all the users

configured-application

HTTP Method	URI	Description
POST	/configured-application/	Creates a configured application
GET	/configured-application/{application-id}	Returns a configured application by application-id
GET	/configured-application/count	Returns the number of configured applications in the controller
GET	/configured-application/{start}/{end}	Returns all configured applications in the given range, start & end are 32 bit numbers, start <= end
GET	/configured-application/	Returns all configured applications
PUT	/configured-application/	Updates the configured application (or a list of applications) with the {application-id} and attributes specified in the Request Object
DELETE	/configured-application/{application-id}	Deletes a configured application by application-id
DELETE	/configured-application/	Deletes all configured applications stored in the controller

policy

HTTP Method	URI	Description
POST	/policy	Creates a policy
GET	/policy/{policy-id}	Returns a policy
GET	/policy/count	Returns the number of policies
GET	/policy/{start}/{end}	Returns all the policies stored in the given range, start & end are 32 bit numbers, start <= end
GET	/policy/	Returns all the policies
PUT	/policy/	Updates the policy attributes for policy-ids specified in the Request Object
DELETE	/policy/{policy-id}	Deletes a policy by policy ID
DELETE (To Be Implemented)	/policy/{policy-name}	Deletes policy by policy name
DELETE	/policy	Deletes all the policies stored in the controller

qos

HTTP Method	URI	Description
POST	/qos	Enables EasyQos for given scope & mappingName as a Qos Status object (scope (default = all), mappingName,

		status)
DELETE	/qos/scope/{scope}/mapping-name/{mappingName}	Disables EasyQos for given scope (default = all) & mappingName
POST	/qos/policy/scope/{scope}	Save the App-Class-Map into the DB & create the marking policy on the devices for the given scope (default = all) (DB + Network changes). Call this API only when you know this mapping is "enabled" otherwise call the '/qos/app-class-map/' POST Api
DELETE	/qos/policy/scope/{scope}/mapping-name/{mappingName}/app-name/{appName}	Remove the App-Class-Map from the DB & delete the marking policy from the devices for the given scope (default = all) (DB + Network changes). Call this API only when you know this mapping is "enabled" otherwise call the '/qos/app-class-map/mapping-name/{mappingName}/app-name/{appName}' DELETE Api
GET	/qos/status	GETS all Qos Status
GET	/qos/status/scope/{scope}	GETS Qos Status for this particular {scope}
GET	/qos/status/mapping-name/{mappingName}	GETS Qos Status for this particular {mappingName}
GET	/qos/class	GETS all Qos Classes
GET	/qos/app-class-map	GETs all Mapping info (Mapping, Applications & Classes)
GET	/qos/app-class-map/mapping-name	GETs DISTINCT mapping names for App-Class-Maps
GET	/qos/app-class-map/mapping-name/{mappingName}	GETs Mapping info (Mapping, Applications & Classes) for this particular {mappingName}
POST	/qos/app-class-map	Creates a new Mapping info (Mapping, Applications & Classes) (1 or many objects in the same request)
DELETE	/qos/app-class-map/mapping-name/{mappingName}	DELETes all mapping info (Mapping, Applications & Classes) for this particular {mappingName} ('cvd' can never be deleted)
DELETE	/qos/app-class-map/mapping-name/{mappingName}/app-name/{appName}	DELETes the mapping info (Mapping, Applications & Classes) for this particular {mappingName} & {appName} ('cvd' can never be deleted)

qos-compliance

HTTP Method	URI	Description
POST	/reference-config	Upload the reference config the type. The type could be a name to identify the config uploaded.
PUT	/reference-config	Uploads a new config file with the existing config name
GET	/reference-config	Returns all the reference configs uploaded.
DELETE	/reference-config/{config-name}	Deletes the reference-config named {config-name}
POST	/qos/compliance/	Do QoS compliance check on the given device IDs or the scope tag (given in the request object) with the reference config The scope tag could be in the location of the device, device-role of the device, or an actual tag
GET	/qos/compliance/{job-id}/{device-id}/status	Returns the status of compliance check for the given ID as Failed / In progress / Completed
GET	/qos/compliance/{job-id}/{device-id}/result	Returns the result of compliance check for the given ID as Compliant / Non-Compliant
GET	/qos/compliance/{job-id}/{device-	Returns the compliance diff for the given ID

	id}/diff	
POST	/qos/compliance/fix	Initiates QoS compliance fix for the given device IDs
GET	/qos/compliance/fix/{job-id}/{device-id}/status	Returns the status of QoS compliance fix

tftp-server-configuration

HTTP Method	URI	Description
POST	/tftp-server-configuration/	Creates a tftp-server-configuration record, write the properties file and tftp server configuration file
GET	/tftp-server-configuration/	Returns a tftp-server-configuration record
PUT	/tftp-server-configuration/	Updates the tftp-server-configuration record, write the properties file and tftp server configuration file
GET	/tftp-server-configuration/default-ip	Returns default tftp-server ip addresses

configuration-file

HTTP Method	URI	Description
POST	/configuration-file/	Creates a configuration-file
GET	/configuration-file/{configuration-file-id}	Returns a configuration-file
GET	/configuration-file/count	Get the number of configuration files.
GET	/configuration-file/{start}/{end}	Returns all the configuration-files in the given range, start & end are 32 bit numbers, start < end
GET	/configuration-file/	Returns all the configuration-files
PUT	/configuration-file/	Updates the configuration-file record with the attributes for configuration-file-id specified in the Request Object
DELETE	/configuration-file/{configuration-file-id}	Deletes a configuration-file
DELETE	/configuration-file/	Deletes all configuration-files stored in the controller

image

HTTP Method	URI	Description
POST	/image/	Creates an image
GET	/image/{image-id}	Returns an image
GET	/image/count	Returns the number of images in the controller
GET	/image/{start}/{end}	Returns all the images stored in the given range, start & end are 32 bit numbers, start < end
GET	/image/	Returns all the images
PUT	/image/	Updates the image record with the attributes and image-ids specified in the Request Object
DELETE	/image/{image-id}	Deletes an image
DELETE	/image/	Deletes all images stored in the controller

ztd-rule

HTTP Method	URI	Description
POST	/ztd-rule/	Creates a ztd rule
GET	/ztd-rule/{ztd-rule-id}	Returns a ztd rule
GET	/ztd-rule/count	Returns the number of ztd rules
GET	/ztd-rule/{start}/{end}	Returns all the ztd rules stored in the given range, start & end are 32 bit numbers, start < end
GET	/ztd-rule/	Returns all the ztd rules
PUT	/ztd-rule/	Updates the rule with the attributes and ztd-rule-ids specified in the Request Object, if "ztd-ulre-id"="ALL", update all
DELETE	/ztd-rule/{ztd-rule-id}	Deletes a ztd rule
DELETE	/ztd-rule/	Deletes all ztd rules stored in the controller

unclaimed-device

HTTP Method	URI	Description
GET	/unclaimed-device/count	Returns the number of unclaimed devices in the controller
GET	/unclaimed-device/{start}/{end}	Returns all the unclaimed devices stored in the given range, start & end are 32 bit numbers, start < end
GET	/unclaimed-device/	Returns all the unclaimed devices
POST	/unclaimed-device/device-add/	Pushes the image and config to the device specified by device_id in the request object
DELETE	/unclaimed-device/{unclaimed-device-id}	Deletes an unclaimed device
DELETE	/unclaimed-device/	Deletes all unclaimed devices stored in the controller

topology

HTTP Method	URI	Description
GET	topology/L2/{vlanId}	Get L2 topology data
GET	topology/l3/ospf/	Get L3 topology for OSPF protocol
GET	topology/l3/isis/	Get L3 topology for ISIS protocol
GET	topology/physical/	Get physical topology data
GET	topology/routing-path/{src-host-ip}/{dest-host-ip}	Get topology routing path
GET	topology/status/	Get topology status
POST	topology/status/	indicate topology that status of network has changed

acl

HTTP Method	U	Description
GET	acl/device/{network-device-id}	Get All ACL from device
GET	acl/interface/{interface-id}	Get ACL from interface

GET	acl/conflict/{acl-id}	Find Conflict ACEs for specified ACL ID
POST	acl/interfaces/	Create a cached ACL list by interface-ids posted and return the list
POST	acl/devices/	Create a cached ACL list by network-device-id posted and return the list
POST	acl/trackacls/	Craete a track between hosts to find is the specified application is passed and return the relevant ACEs

energywise-info

HTTP Method	URI	Description
GET	/energywise-info	Get all energywise objects
GET	/energywise-info/network-device/{network-device-id}	Get energywise objects by network device