Solution Architecture

Revised: May 23, 2022

Back

Solution Architecture

This chapter provides insight into how the orchestration engine processes a service request, using service definition models to create and instantiate services.

This chapter contains the following sections:

Modular Architecture

As Cisco MSX evolves, the architecture continues to modularize components creating clear demarcation points between various layers in the solution, allowing maximum flexibility in both commercial and deployment models.

Cisco MSX Modules

- *Service Interface (SI)—Customer Facing Level*

  The Service Interface consists of two modules, a customer facing Front-End and service request processing Back-End.

  The SI Front-End is the Self-Service User Interface; from here the end-customer can select the service offer which meets their requirements. At that point, the end-customer can construct Cisco MSX services consistent with the service offer.

  The SI Back-End is separate module that constructs the service request based on the end-customer choices in the Front-End module.

- *Cisco MSX Platform—Resource Facing Level*

  In cases where Cisco MSX is the device configuration controller, Network Service Orchestrator provisions services based on Service Packs that logically mirror Customer Facing Service (CFS) level packages. Service packs are internal software modules that house the service models execution logic that define specific Service Packages.

  In cases where Cisco MSX interacts with a device controller, Cisco MSX will leverage the API(s) provided by that controller to instantiate the appropriate outcome requested by the end-customers service request.

Cisco Network Services Orchestrator

Cisco Network Services Orchestrator (NSO) is a model-driven orchestrator which uses YANG for modeling services, and can use various methods such as NETCONF, SSH, and REST-based APIs to provision the devices. NSO receives a service intent request through the open API interface presented northbound to Cisco MSX.

Network Element Driver (NED) software modules are used to abstract out the different physical and virtual network devices to which service configuration data may be pushed. NEDs allow the same service definition models to apply to equipment from multiple vendors.

NSO runs as a container in the Cisco MSX solution. The NSO container is deployed in one of the Kubernetes nodes and monitored by the Kubernetes control plane. Service packs which utilize NSO have their own NSO container.

Cisco NSO as the Plug-and-Play Server

The Plug-and-Play (PnP) Server software element is used to handle and achieve zero-touch deployment (ZTD) for CPEs coming online and wanting to utilize the services configured through Cisco MSX. In the Cisco MSX solution, NSO also functions as the PnP server. Once a CPE device is connected to the Internet, a “call home” protocol communicates with the Cisco MSX PnP Proxy Agent. The Proxy agent is responsible for finding any NSO complexes that have registered to serve that device. Since we can only have a device registered to one NSO, if one is found, the request is forwarded to NSO. If no, NSO has registered for the device, that device is sent a BACKOFF message to try and connect again at a future time. The CPE device is fully configured using a four-step process:

- Day-(-1)—Initial config of CPE to find the PnP Server

Note A new device shipped from Cisco has no knowledge of where or how to contact its PnP server. The device first reaches out to http://software.cisco.com to receive the day-(-1) configuration. After the initial configuration, a PnP redirect provides the PnP server configuration hosted by the operator.

- Day-0—Configuration of the device management interface
- Day-1—Basic interface and system configuration
- Day-2—Service-specific configuration

Inclusion of the PnP service in the Cisco MSX solution removes costly truck-rolls, which are traditionally required to install and configure each CPE device, from the service deployment. The removal of this activity greatly reduces the cost-of-service deployment, a cost that has traditionally been a pain point for Operators.

Virtual Infrastructure Manager

The Cisco MSX platform leverages or Amazon Web Services as the Virtual Infrastructure Manager (VIM) for its deployment. Cisco MSX is constructed as a collection of Docker containers that are deployed and managed using a Kubernetes cluster. This approach is used to define an open platform which facilitates automated deployment, provides scaling, and simplifies operation of containers. Cisco MSX uses Ansible to deploy and manage the construction of the basic infrastructure followed by deployment of the platform and service pack software.

Service Interface

The Cisco MSX service interface captures the service intent of the customer and realizes this intent by interacting with NSO or other domain specific controllers. The service interface is composed of two subsystems: the web portal (front-end) and the back-end.

The web portal is designed as an all-in-one web-based solution GUI. Based on the user login type, the user will be presented with one of three service interfaces: administer, operator, or user. Upon successful authentication, the user is directed to the appropriate web interface, based on the predefined role of the user.

The service interface, based on the user's role, will allow administrators to provision a tenant space for end users, while the operators can view the status of all services running, and lastly the user role is for the end-consumer to order the service based on their requirements.

From Cisco MSX 3.10 release onwards, the Cisco MSX portal displays new GUI.

The Cisco MSX GUI has:

- **The Operator Workspace:** Only visible to operator users. It lists all tenants that the operator is managing and the services they have subscribed to.

  Click on a tenant's tile to see details specific to a tenant in the Tenant Workspace GUI.

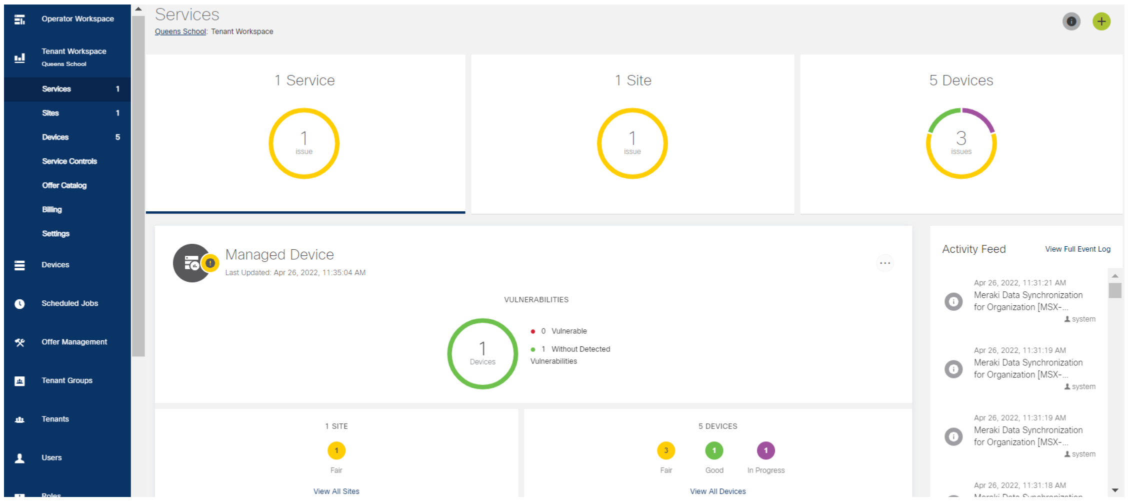
- **A Tenant Workspace:** Allows tenants to access the information related to their subscribed services.

  The menus that are available in the Tenant Workspace are:

- **Services**: Display all services subscribed by a tenant, service status, and other service metrics.
- **Sites**: Display an overview of the tenant's sites, site status, and allows access to site details.
- **Devices**: Display an overview of the tenant's devices, device status, and allows access to device details.
- **Service Controls**: Display the custom service controls that are used to manage the services.
- **Offer Catalog**: Display existing subscriptions and allows subscribing to new services.
- **Billing**: Display billing information about the tenant's subscriptions.
- **Activity Feed**: The Cisco MSX portal allows a tenant to view several events pertaining to the subscriptions, sites, devices, templates, and services. The events that are logged in the **Events Log** window are also used in the Activity Feed. To view the Activity Feed, choose **Tenant Workspace** > **Services**. These contextual event feeds are also displayed on the **Sites Overview** window and **Devices Overview** window.
- For more information on monitoring service status, see the Cisco Managed Services Accelerator (MSX) 4.3 Administration Documentation.

The figure below shows the Tenant Workspace:

Figure 1: Tenant Workspace



The back-end is the composition of micro-services that together communicate with various components in the Cisco MSX solution. The back-end processes the service request input using the web interface or self-service portal, and creates parametrized service requests to NSO or domain controllers.

The Cisco MSX services that are available through the portals are dependent on the service packs made available by the operator.

The service interface back-end communicates with the web portal through a REST-based API gateway. The pages of the web portal rely on back-end micro-services to process user data entered in the various portal screens. Depending on the type of data entered, the information will be sent to/from the back-end API and delivered to/sent from the micro-service responsible for processing the incoming data. The back-end micro-services are responsible for multiple functions as described in Chapter 2.

Back