Architecture for Lifecycle Service Automation: Cisco Network Services Orchestrator Enabled by Tail-f
Contents

Executive Summary 3
The Automation Opportunity 3
What Does End-to-End Service Orchestration Look Like? 4
Cisco Network Services Orchestrator:
Supporting Full Lifecycle Automation of Network Services 4
Pillar 1: Data Models and Data Model Mapping 4
Pillar 2: State Convergence 4
Pillar 3: Orchestration Across Multiple Domains 5
Pillar 4: Orchestrated Assurance 5
A Platform for On-Demand Lifecycle Service Orchestration 5
Network Services Orchestrator Architecture Overview 6
Data Models and Mapping 8
Orchestrating with Precise Data Models 8
State Convergence 9
State Convergence with Service Mapping 9
State Convergence in Action 10
Orchestrating Across Multiple Domains 11
Multi-Domain Orchestration in Action: NFV MANO 11
Multi-Domain Orchestration in Action: SDN 13
Orchestrated Assurance 14
Orchestrated Assurance with Network Services Orchestrator 14
Implementing Orchestrated Assurance 15
Design-Time Assurance 16
Run-Time Assurance 16
Cisco Network Services Orchestrator in Action:
Orchestrating Service Fulfillment with Assurance 17
Designing Activation Testing Models 18
Modeling Ongoing SLA Monitoring 19
Instantiating Activation Testing and Assurance 20
Orchestrating Service Assurance Functions 21
Service Orchestration with Assurance in the Network Services Orchestrator User Interface 22
Conclusion 23
Executive Summary
The network services landscape has never been more competitive. New technology innovations, especially around network functions virtualization (NFV), are increasing customers’ expectations of network service providers. The winners will be those that are most agile—able to design new services faster, customize them to each customer’s needs, and deliver them at the speed of software. The secret to achieve these goals is service orchestration. But in complex and dynamic operator environments, it’s not enough to orchestrate one part of a service or automate functions within a single network domain. True agility demands orchestration across the full service lifecycle.

Cisco® Network Services Orchestrator enabled by Tail-f provides the platform to make end-to-end service orchestration a reality. This paper describes:

• The four critical characteristics that a service orchestrator must have to automate full lifecycle services in modern operator networks
• The Cisco Network Services Orchestrator reference architecture, with details on how it puts these principles into practice
• A use case illustration that demonstrates the full lifecycle orchestration of a real-world service

The Automation Opportunity
The evolving network services marketplace promises huge new opportunities, and significant challenges. The combination of NFV, software-defined networking (SDN), and self-service ordering from the cloud hold enormous potential for operators to deliver a broader range of customized services, more quickly, to more markets. But these innovations are also driving high customer expectations.

Customers are now accustomed to purchasing a wide range of technology products online. They expect to be able to configure products and services to get exactly what they want, and when and how they want it. And they see no reason why network services shouldn’t offer the same kind of experience.

The day is quickly approaching where network service providers that want to be competitive will need to be able to compose customized products on demand, and deliver them in minutes or even seconds.

In fact, a recent survey of 160 global operators conducted by Heavy Reading revealed that the top priority for network service providers by far is streamlining the design and delivery of new services.

Unfortunately, the same technology trends driving customer expectations, especially network virtualization, have made service design and delivery vastly more complex. If service providers are to become as fast and flexible as their customers demand, they will need to move away from a model based on complex service interdependencies and hard-coded integration. Instead, they need a framework in which all possible network services can be viewed as atomic functions that can be assembled and disassembled in myriad ways, on demand. They need automated, repeatable, reliable, end-to-end service orchestration.
What Does End-to-End Service Orchestration Look Like?

In this context, orchestration entails the full lifecycle automation of service activations and changes in the network. It implies the programmatic configuration of all elements participating in a service—not just virtualized functions—in a single transaction.

That automated configuration needs to extend from service-level parameters through the automated configuration of every network device involved in the service. It can’t be limited to a single domain, as real-world network topologies are hybrid, and likely to remain so for the foreseeable future. It should encompass both physical and virtual network elements, SDN and traditional network management interfaces, and conventional OSS systems as well as NFV orchestrators. Finally, it needs to address the entire application lifecycle of the service, including fulfillment, activation testing, and ongoing monitoring and assurance.

This last point is particularly important in an increasingly virtualized world. If customers are going to be able to order and activate services on demand, assurance can’t be an afterthought. There should be no delay while the assurance aspects of a service are ingested and mapped to a new management model by a separate system. Assurance must become part of the service fulfillment process, driven by an overarching orchestration system, and using the same service models.

Cisco Network Services Orchestrator: Supporting Full Lifecycle Automation of Network Services

Cisco Network Services Orchestrator enabled by Tail-f provides end-to-end automation to design, deliver, and assure services faster. It provides a single, network-wide interface to all network devices and services, both physical and virtual, using a common modeling language and data store. It lets operators define services using the standardized YANG modeling language, map those services to YANG device models (as well as non-YANG devices), and automate everything, from high-level intent to granular device configurations. So operators can easily create and change services using standardized models, without lengthy custom coding or service disruptions.

Network service providers can automate a wide range of multivendor devices across physical and virtual environments. They can orchestrate the entire service lifecycle, including fulfillment, activation testing, and ongoing service-level assurance, through a single transaction and a single data model. And they can continually refine and repackage network services at the speed of software. (For an overview of Network Services Orchestrator capabilities and benefits, visit http://www.cisco.com/go/nso.)

Network Services Orchestrator delivers these capabilities by embracing what Heavy Reading calls the “four pillars” of successful service orchestration. These pillars describe the key characteristics that a service orchestrator must provide to automate the full lifecycle of a service in today’s increasingly complex, dynamic, and virtualized operator environments. They are: data models and data model mapping, state convergence, orchestration across multiple domains, and orchestrated assurance.

Pillar 1: Data Models and Data Model Mapping

The ability to use common data models and data model mapping to describe all services and devices is the foundation of automating network configuration tasks. Network Services Orchestrator allows operators to model all services in a precise and semantically rich way using the standardized YANG data modelling language.

By precisely describing both service-level intent and device-level configurations in a common language, Network Services Orchestrator can manipulate data models programmatically. Which means that operators can map service models to device models declaratively, without manual coding. They can design, fulfill, and assure services faster, in an entirely automated way. And they can do it with much greater flexibility than traditional approaches that rely on hard-coded service logic and endlessly proliferating workflows.

Pillar 2: State Convergence

To achieve true end-to-end automation, the orchestrator should be able to receive the “intent” of the service (or the black box input parameters of the desired service or change), and translate that to real change in the network. Many operators currently rely on workflow definitions to accomplish this, and quickly find themselves overwhelmed by a constantly growing body of workflows to account for each unique case.
Network Services Orchestrator takes a different approach to service lifecycle management using the concept of state convergence.

Using the same common data models and modeling language to describe services and devices, Network Services Orchestrator fully automates the creation, modification, and deletion of network services end to end. It maps design-time service definitions to run-time network operations programmatically, through a single, flexible data model for a service. And this data model mapping becomes an iterative process. Here’s how it works.

Network Services Orchestrator uses a run-time state convergence algorithm that requires just one workflow definition: here’s what we need this service to do. At run-time, the state convergence algorithm examines the available network resources and status, calculates the difference between the current state and the new input, and finds the shortest possible path to achieve that desired state.

The same thing happens whenever anything changes in the network, whether due to instructions from northbound systems or changes in the underlying devices. Network Services Orchestrator looks at the model describing what the service needs to be doing, and dynamically reconfigures the network to fulfill the desired service intent. All of this happens in minutes or even seconds, at run-time.

By using this state convergence approach, rather than pre-defining a never-ending series of workflows to account for every possible change, Network Services Orchestrator dramatically reduces overhead for network service orchestration. At the same time, it increases reliability during service activations and post-deployment changes, both of which reduce service delivery times and costs.

**Pillar 3: Orchestration across Multiple Domains**

Many orchestration solutions are designed exclusively for SDN or NFV. But in most real-world operator environments, virtualized network functions (VNFs) are just one part of a full customer-facing service. To achieve true agility and speed, operators need to automate network services end to end.

Network Services Orchestrator fulfills all of the requirements for NFV service orchestration as defined in the NFV Management and Orchestration (MANO) stack. As part of a single service model, Network Services Orchestrator can trigger the launch, configuration, and ongoing monitoring and license management of VNFs, both individually and in complex service chains.

It can also automate configuration and monitoring across the traditional network management domains and physical devices that still make up a significant part of operator environments. Operators can automate the full end-to-end service in a single transaction, helping to enable faster delivery times, lower operating costs, and self-service ordering on demand.

**Pillar 4: Orchestrated Assurance**

Automating the full lifecycle of a service means going beyond orchestrating provisioning and accounting for the ongoing monitoring and assurance of the service. Network Services Orchestrator lets operators configure service activation testing, ongoing monitoring, and service-level agreement (SLA) assurance functions synchronously with fulfillment—once again, using the same common data model.

Operators can define run-time key performance indicators (KPIs) as part of the service model, and configure service monitoring devices to measure them from the right location. They can draw on the orchestration system’s real-time, stateful view of the network, and automate the instantiation of virtual probes in the network. They can measure the customer experience of the service directly, both at activation and on an ongoing basis, rather than inferring it from indirect device-level metrics. And they can use Network Services Orchestrator state convergence capabilities to automatically respond to any issues detected, redeploying services as needed to maintain SLAs. All of this translates to reduced operational overhead for operators, fewer errors and truck rolls, and better experiences for customers.

**A Platform for On-Demand Lifecycle Service Orchestration**

If operators are going to meet customer demand for faster, on-demand service provisioning in an NFV world, orchestration systems of the future must address each of these four principles. Network Services Orchestrator delivers them today, not in a product roadmap, but in running code that can be demonstrated in operator networks.

The following sections provide an overview of a Network Services Orchestrator reference architecture that encompasses the four pillars of successful orchestration, and illustrate how it implements them. Finally, this paper provides a use case illustration of the Network Services Orchestrator architecture provisioning an end-to-end service with orchestrated assurance.
Network Services Orchestrator Architecture Overview

Network Services Orchestrator is now widely deployed and proven as an orchestrated fulfillment solution in major operator networks worldwide. But as operators know well, there are many aspects of orchestrating a service beyond provisioning.

With the recent release of Network Services Orchestrator 4.1, Cisco can provide a comprehensive reference architecture to accomplish end-to-end, multi-domain service automation (Figure 1). This includes full lifecycle orchestration of a service, including fulfillment, assurance, and NFV orchestration and service chaining in accordance with the MANO framework. And it encompasses both physical and virtual network devices, from both Cisco and third-party providers.

Figure 1. Cisco Network Services Orchestrator High-Level Architecture

Northbound of Network Services Orchestrator, Figure 1 depicts the more traditional OSS components that are part of any customer-facing service, including order management systems, customer portals, and event management alarm and fault systems. These are not technically part of the reference architecture, but Network Services Orchestrator inter-operates with them as part of end-to-end service orchestration.

Figure 2 provides a more detailed picture of the assurance, fulfillment, and NFV orchestration elements of Network Services Orchestrator, as they might appear in a real-world operator network. Note that, as shown here, Network Services Orchestrator is modular and open. This reference architecture combines Cisco assets for VNF management, orchestration, and physical device elements, as well as third-party open devices and open-source components. Which means that operators can add and change pieces to best suit their environment, and continually evolve the platform over time.
Here you can see:

- **NFV orchestration:** The NFV orchestration (NFV-O) component of Network Services Orchestrator integrates with the VNF Manager (in this case, the Cisco Elastic Services Controller) to onboard VNFs. It can draw on widely used open-source NFV components such as OpenStack virtual infrastructure manager (VIM) and Open vSwitch, as shown here, to create the necessary service chains among VNFs and physical network elements.

- **Orchestrated fulfillment:** In the middle, Figure 2 shows the service orchestrator that provides the core functionality of Network Services Orchestrator. It translates the high-level intent for the networking service into device configurations, and pushes those configurations down to the physical and virtual devices that encompass the service. As an open platform, Network Services Orchestrator can orchestrate devices from various vendors. In this illustration (which could support the kind of Layer 3 VPN use case detailed later in this paper), Network Services Orchestrator can create service chains using virtual routing, switching, and firewall devices from Cisco, as well as third-party virtual firewalls.

At the same time, it can also configure physical devices, including routers from Cisco, Juniper, and Alcatel-Lucent.

- **Orchestrated assurance:** Network Services Orchestrator on its own is not an assurance system. However, as part of configuring or instantiating a service in the network, it also configures the assurance systems so that they become service-aware and active. Once again, Network Services Orchestrator can incorporate third-party systems to accomplish this. When activating a service, for example, Network Services Orchestrator can synchronously activate performance monitoring capabilities, including:
  - Instantiating virtual test probes to perform activation testing and provide ongoing measurement of service KPIs against defined SLAs for a service
  - Maintaining the service topology in an assurance or inventory system, which can enable accurate service impact and root-cause analysis (for example, using the “ZenOSS Dynamic Service View”)
  - Instantiating IP SLA monitoring and management applications that use metrics engines or Cisco Unified Communications Session Management Edition for voice and video sessions
All of these elements—from onboarding and instantiating VNFs, to configuring and activating a network service, to defining SLAs and driving assurance functions—are controllable through the Network Services Orchestrator user interface. It also includes NFV orchestration capabilities, including tools to design and build network service descriptors. And the architecture integrates with northbound assurance systems, such as Incident.MOOG for fault analysis, as well as other conventional OSS components.

Let’s take a closer look at how this architecture embodies the four pillars of successful orchestration.

Data Models and Mapping

Network Services Orchestrator is unique in that it uses formal languages to define all data models for services. Everything that Network Services Orchestrator orchestrates—whether fulfillment or assurance, virtual or physical function—is first modeled in YANG data models. Network Services Orchestrator orchestrates entirely based on those abstracted data models, rather than domain-specific information about the service. Why is this so important?

In conventional service activation, there is a significant gap between the informal models used to describe a service at a high level, and the fine-grained command-line interfaces (CLIs) and APIs used to configure the actual devices. This gap represents the biggest barrier to fast, efficient lifecycle automation of network services.

At the higher level, domain experts define services using Visio diagrams, Word documents, Unified Modeling Language (UML) diagrams, or other informal means. These models are useful for human-to-human discussions about planned services. But they can’t be understood programatically by an orchestration system, which means they can’t be automatically implemented.

This is good news for professional services teams, who embark on long, expensive implementation projects to translate informal service definitions into workflows that instantiate the service across the CLIs or APIs of a specific set of devices. But it’s not so good for operators, who must spend a lot more time and resources to introduce a new service. And this translation process is not a one-time engagement; new workflows will be needed for every new service, and every operation that needs to be carried out on the service or the devices implementing it.

Additionally, when an operator’s domain experts use this approach, they are handing over the interpretation of service intent to a software implementation team—meaning they don’t know or control the real semantics of the service they’re modeling. So their response to changing customer demands will be inherently slower, more cumbersome, and potentially higher-risk.

Orchestrating with Precise Data Models

Now, consider what can be achieved when product teams use concrete, semantically rich data models for services (Figure 3). Note that this assumes a standardized modeling language, such as YANG, with a well-defined grammar that can be parsed by a compiler. And it assumes that the operator uses formal data models in the same language for devices—both physical and virtual—with precise schema that details device capabilities.

Figure 3. Orchestrating with Common Service and Device Data Models

First, once operators have text-based, standardized data models for services and devices in the same language, they can fully automate and render them, without the intermediate data transformation project. Similar to the way programmers used to code in Unix, it now becomes possible to use automated tools and machine reading to parse any element of a service and pass it on to the next process.

Operators can now use declarative data model mapping to move from high-level service intent to granular device configurations—the foundation for fast, automated service delivery. The same service models can be reused over and over again, even when applying them to different customers, locations, and devices—from multiple vendors, both physical and virtual.
And, since orchestration is entirely data-model-driven, the full semantics of the service are also now controlled by domain experts, instead of professional services teams. Massive data transformation projects are no longer needed. Services can be created and updated more easily. And operators gain the agility to respond to new customers and opportunities faster.

**State Convergence**

Even as modern orchestration systems have embraced data models, there remains a gap between defining service data models at design-time and implementing them in the network at run-time.

The conventional answer has been to use workflows to bridge that gap. That is, every step in the provisioning process is expressed through a series of workflows defining fine-grained configuration details for a specific set of devices (and potentially, for a specific customer).

In practice, however, a workflow-based approach is not precise or flexible enough to keep pace with complex, real-world operator environments. Even when operators employ reusable workflow templates, defining all of the necessary workflows—detailing how to deploy, modify, and delete each service element to achieve a specific final state—becomes a never-ending task (Figure 4).

**Figure 4.** Orchestrating without State Convergence

![Diagram showing the lack of state convergence](image)

- Endlessly proliferating workflows
- Complexity grows exponentially
- Orchestrator focuses on how to achieve final state

For example, an operator may create a data model at design-time for a Layer 3 VPN, incorporating device data models for all the vendors in the network. It may seem intuitive to simply create a workflow to transform that modeled service instance into precise device configurations. But what about customers who need Lightweight Extensible Authentication Protocol (LEAP) support? Create another workflow. What if they want to add a new leg to their VPN? Another workflow. What about changing cost parameters? Another workflow.

In a typical multivendor operator environment, the complexity of workflows grows exponentially. And that’s just writing new workflows for each possible permutation of a service. Maintaining consistency across all of these sprawling workflows as the environment changes becomes its own enormous task. This is to say nothing of the complexity associated with virtualization, where southbound virtual devices may be scaling and healing spontaneously.

To achieve true agility, operators need to automate any possible change in the service, including both changes to network devices and instructions from northbound APIs, such as from a cell service portal or order manager. Users, and potentially customers, should be able to drive fine-grained network changes (to continually turn up, modify, and tear down services at run time) with simple, declarative commands.

**State Convergence with Service Mapping**

Network Services Orchestrator takes a different approach to make true service automation a reality. Rather than modeling the entire final deployed state of a service in a state machine, it treats deployments as a continuous model-to-model mapping process. Effectively, it takes the high-level service parameters from the operator or customer, and then derives all the steps needed to achieve the desired final state at run time, without requiring the user to provide details about each specific step. It then converges the network toward that desired state automatically (Figure 5).
Think of state convergence as a single workflow definition for the entire service in all its permutations. But rather than providing a step-by-step list of instructions, the workflow describes the high-level intent of the service and the resources available to implement it: “Here is what we need, here are the physical and virtual devices at your disposal, go make it happen.”

For every attribute in the service data model, Network Services Orchestrator identifies the appropriate corresponding value in the device data model. So for example, in a basic Layer 3 VPN, Network Services Orchestrator could model three different possible attributes in the service data model, and a corresponding 25 different possible attributes in the device data models, within a single definition of the service.

Network Services Orchestrator then uses its state convergence algorithm to take that high-level service intent and execute it at run-time in the network. If a northbound system instructs Network Services Orchestrator to create, delete, or modify a service, the algorithm examines the current attributes of the service and network, and derives the appropriate corresponding values for the device data model in seconds. Endless step-by-step workflow definitions are no longer necessary. Instead, the operator retains a single definition of the service, and Network Services Orchestrator pushes the desired intent toward the network automatically.

Note that this state convergence algorithm is not something that’s run once; it is a continual, iterative process that happens automatically in response to every change (northbound or southbound) in the network.

State Convergence in Action

This state convergence approach is essential to full lifecycle service orchestration, because it’s the fastest and most efficient way to automate complex real-world operator environments. Imagine, for example, that as one element of a Layer 3 VPN service, the orchestrator needs to trigger an external system (such as OpenStack) to create a virtual firewall. This can take a few minutes. As Network Services Orchestrator runs its convergence algorithm and attempts to reach the desired end state, it will recognize that the virtual firewall is not yet live. At that point, it will halt provisioning, waiting for that event. Network changes that have already been made as part of the deployment remain in place. But as soon as the firewall VM is booted, this event triggers the state convergence algorithm to rerun from the beginning, verifying that all the previous network changes are still valid.

If they’re not (for example, if the virtual router created in a previous step has gone down), Network Services Orchestrator goes back and fixes that problem, and then reruns the convergence algorithm again (and again) until the final desired state is achieved. Or, in the event an error case cannot be resolved, it rolls back all configurations to the beginning of the transaction, “un-deploying” each step in the service, in the correct sequence, to eliminate any impact on live services.
Central to these convergence capabilities, Network Services Orchestrator uses a “minimum diff” engine that calculates the minimum number of changes necessary to go from the current network state to the desired final state. This applies to all service lifecycle operations: create, modify, delete. Network Services Orchestrator can rerun a deployment as many times as necessary to converge on the final desired state, without affecting network change steps that have previously completed successfully, dramatically reducing system overhead.

Network Services Orchestrator can also trigger state changes to other systems, such as an OpenStack VIM or an IP addressing system. It uses a “publish-and-subscribe” (pub/sub) messaging model to keep tabs on all other elements of the architecture, detect responses to orchestration requests, and rerun deployment instructions to virtual infrastructure or external systems.

Once again, all of this happens continuously, for every northbound or southbound event in the network. It’s this process—using a single data model for services and devices, data model mapping, and iterative state convergence—that can enable full automation without workflows.

Orchestrating across Multiple Domains

“End to end” is one of those terms used a great deal by technology vendors, but what does it actually mean? From the perspective of a network service provider, it’s about having a single orchestration system that can fulfill every aspect of a service, whether networking elements are physical or virtual, and whether they’re interfaced through an SDN controller, an NFV orchestrator (NFV-O), or traditional CLI.

A complete customer-facing service often involves some SDN, some virtual functions, a great deal of classical network configuration, and finally, applications and assurance (Figure 6). So any orchestration system that addresses only part of the landscape cannot truly automate services end to end.

This is particularly important to remember with current industry marketing around NFV orchestration. VNFs may provide important elements of a service, and the NFV-O, in activating and service-chaining virtual functions, plays a critical role in service orchestration. But revenues are not driven from creating virtual machines—they’re driven from end-to-end services, which are inextricably tied to operators’ OSS/BSS and element management systems (EMS). Even if VNFs compose some or most of a service’s infrastructure, the customer- and service-specific configurations that are the core of an operator’s business require orchestration that extends beyond the NFV domain.

Figure 6. Orchestration Spans Multiple Domains

Network Services Orchestrator provides an overarching orchestration system that resides above the NFV-centric elements. From there, it can automate the full end-to-end application lifecycle of a service, encompassing the NFV MANO system, SDN controllers, and conventional OSS, BSS, and EMS elements. Let’s take a closer look at how this works.

Multi-Domain Orchestration in Action: NFV MANO

As discussed, the NFV MANO stack manages only the activation and assurance of VNFs, individually or in service chains of network applications. (See sidebar.) It can’t address those aspects of a customer-facing service that rely on physical devices, which still make up a large part of most operator environments.

Adding a further challenge, once VNFs are activated, most hybrid operator environments treat them the same way they would treat physical network devices. That means, for example, that virtual devices must support multitenant services. (A single virtual aggregation router, for instance, should be able to serve multiple customers in the same way a physical aggregation router can.)

Clearly, automating a customer-facing service end to end, even when the service is composed of VNFs, will require additional OSS configuration beyond the scope of NFV MANO (Figure 7).
Untangling Network Services and Virtual Network Applications

Some confusion has arisen in the industry around what the term “network service” means for NFV MANO versus for a network service provider.

When the European Telecommunications Standards Institute (ETSI) references a network service in the NFV MANO stack, it’s referring to the day-zero configuration of VNFs and VNF service chains. (The same is true for Topology and Orchestration Specification for Cloud Applications [TOSCA], where network service means a set of booted applications.) That can consist of an individual VNF, or a set of VNFs chained together to provide a given application, such as a virtual IP multimedia subsystem (vIMS) or virtual evolved packet core (vEPC).

Of course, operators define a network service very differently. To service providers, network service refers to a higher-level service offering (E-Line, VPN, 4G mobile data, etc.) that spans multiple network elements and applications. And a single network element may support multiple service instances.

From an operator’s perspective, even an operationally-ready vIMS or vEPC instance is just a subset of an actual service. For clarity then, this paper uses the term network service in the standard way that operators understand it, and refers to VNFs and service chains in the NFV MANO stack as network applications.

Figure 7. NFV Orchestration

Cisco NSO provides an overarching lifecycle automation framework. It handles the final service configuration, taking on the full NFV-O role within the NFV MANO framework while also incorporating OSS functions. It communicates directly with the VNFs in the service to perform run-time configuration, trigger VNF instantiation, and set up service chains (Figure 8).

As defined in the NFV MANO specification, Network Services Orchestrator can also support predefined templates, such as MANO descriptors or TOSCA templates, to manage the cloud aspects of VNF and network application lifecycles. This includes configuring an application instance for day-zero operational readiness; installing, uninstalling, and reinstalling a network application instance; and starting or restarting an application instance (for example, to implement a VNF configuration change).

In this way, Network Services Orchestrator provides full NFV-O functionality for virtualized applications. And it delivers it within an overarching orchestration framework that goes beyond NFV MANO to automate the complete application lifecycle of customer-facing services.
Figure 8. Orchestrating NFV across the Application Lifecycle

Multi-Domain Orchestration in Action: SDN

Network Services Orchestrator can also address elements of a customer-facing service driven by an SDN controller. In a typical use case, an operator uses SDN when activating VNFs in the data center. The SDN controller is responsible for stitching the VNFs together into the desired network application and connecting the data center to the network.

In these scenarios, Network Services Orchestrator is still the source of the end-to-end service models.

But instead of triggering the NFV-O or directly interfacing with physical devices in the network, it triggers the SDN controller to activate the required physical and virtual network functions (Figure 9).

Once again, the key takeaway is that, while SDN functions are extremely important, they still represent a subset of the complete customer-facing service. And Network Services Orchestrator automates them as part of an end-to-end orchestration framework that encompasses the full service lifecycle.

Figure 9. Orchestrating with SDN
Orchestrated Assurance

Assurance calls to mind a variety of complex systems and testing instruments. But in the end, they all come down to one question: Is the customer getting the quality of experience (QoE) they paid for? Unfortunately, in many operator environments, answering this question is no easy task.

Part of the problem is that service fulfillment and service assurance have traditionally been handled as two separate processes, managed through separate systems, at different times, often by completely different operational teams. In many operator environments, the fulfillment team’s role ends once the circuits are activated and confirmed to be live. It’s then the assurance team’s job to go through the arduous and often expensive process (especially for new services) of collecting network data and verifying that the service is performing as expected.

As operators move toward an on-demand, self-service system of network services, they will need to get better at both assuring that services are performing as they should be, and accelerating the service assurance process. Three major barriers stand in the way:

• Lack of thorough activation testing: Ideally, operators should be able to verify that every service deployment is performing as expected, at least when activated. Unfortunately, this is rarely the case. Service-level testing and measurement traditionally require expensive equipment and significant operational resources, so it is reserved for only the largest, most important customers. For the rest, many deployments are not tested at all, or rely on basic ping tests. These can verify connectivity, but they reveal nothing about the customer’s actual experience. Service reachability, latency, appropriate traffic prioritization, and other key parameters remain unknown. The result is that a not-insignificant number of deployments are not done correctly the first time. Which translates to higher costs for field efforts, delayed time to revenue, and unhappy customers.

• Siloed fulfillment and assurance: Historically separated from fulfillment operations, assurance systems have evolved to be very different than provisioning tools. Where fulfillment systems typically have detailed real-time information about the network and services, assurance systems usually rely on more limited information that may not reflect recent changes or updates.

Even when information is relatively current, traditional assurance tools presuppose a network that’s relatively static—the opposite of the dynamic environments that operators are moving to with SDN and NFV. Typical assurance systems are also not synchronized with provisioning processes. They use different service models and happen after a handoff from fulfillment, inherently introducing delay. This makes them poorly suited for on-demand service delivery in an NFV world, where operators want to be able to provision, assure, and begin generating revenue from a service in minutes.

• Service KPIs inferred rather than measured directly: Capturing accurate service-level KPIs has been an ongoing challenge for operators for decades. As a result, most operators rely on an indirect, bottom-up approach. They collect and correlate huge amounts of information (from device-centric counters, real-time network traps and alarms, offline data in inventory systems), and try to infer the customer experience indirectly. They capture massive amounts of information, but most of it is simply noise. They struggle to contend with virtual functions that can scale up and down dynamically. And the picture they cobble together of service performance ends up being poorly correlated with the real-world customer experience.

To enable fast, on-demand delivery of high-quality services, service fulfillment and assurance operations should be tightly coupled as part of a single, synchronous transaction. Thorough activation tests should be performed automatically for every new service deployed. And operators should be able to see and remediate the real-world customer experience of their services at any time, on an ongoing basis.

Orchestrated Assurance with Network Services Orchestrator

Cisco Network Services Orchestrator integrates full service assurance—encompassing both automated activation testing and ongoing monitoring of service-level KPIs—with the service fulfillment transaction. Using the same service data model and the same YANG modeling language and tools, operators can define appropriate KPIs for each service at design-time. They can then test services synchronously as they’re provisioned, and monitor and remediate them automatically against customer- and service-specific SLAs on an ongoing basis.
With Network Services Orchestrator, operators can orchestrate assurance functions to:

- **Measure service KPIs directly on the wire:** Network Services Orchestrator allows operators to use a variety of Cisco and third-party tools to measure the service experience from the customer’s perspective—at service endpoints—as part of the orchestrated fulfillment transaction. This includes creating customized virtual probes for each customer and service, and placing them in the best location in the network automatically.

- **Automate in-depth activation testing:** As the final stage in the provisioning process, Network Services Orchestrator performs a service-level stress test of the deployed service (not just a basic ping test) to verify that the service is performing the way the customer expects. It generates a “birth certificate” for the service, a best practice recommended by the Metro Ethernet Forum (MEF) for any customer-facing service.

- **Make services assurable from the moment they’re instantiated:** Network Services Orchestrator eliminates the built-in delays associated with handoffs between traditional fulfillment and assurance processes. The moment a service is deployed, it notifies external assurance systems handling root-cause analysis and remediation, and triggers ongoing service monitoring against defined SLAs.

- **Automate remediation of SLA violations:** When a service isn’t performing the way it should be, the culprit in many cases is an incorrect configuration at some point in the service. By linking assurance processes with fulfillment—where the service model has comprehensive real-time information on device configurations—Network Services Orchestrator can immediately recognize and remediate the problem.

If assurance systems detect a service degradation in virtual devices or the NFV infrastructure, Network Services Orchestrator automatically triggers its state convergence algorithm to redeploy VNFs to address the issue. The full service lifecycle defined in the data model—provisioning, activation testing, configuring ongoing monitoring—then proceeds the same way it would for a new service, as part of a closed feedback loop (Figure 10). Network Services Orchestrator can also flag external assurance systems for faults detected in the physical network. The end result: operators can see and address customer- and service-specific performance problems immediately, instead of waiting for a traditional assurance system to recognize that a device-level fault is impacting a service.

**Implementing Orchestrated Assurance**

Figure 10 illustrates how Network Services Orchestrator orchestrates the full lifecycle of the service—provisioning, activation monitoring, and ongoing assurance. As shown, operators can use a single service model to automate design-time and run-time operations in a continuous, closed-loop process.
Design-Time Assurance

The left-side circle illustrates everything the service model encounters during design-time. The first two elements—configuration and service descriptors for virtual functions—are standard in any orchestration system. As discussed, in Network Services Orchestrator, both service and network device descriptors are defined in the same YANG modeling language, within the same data model. However, Network Services Orchestrator service models extend to the assurance phase of the service lifecycle as well, through elements 3 through 5: configuring test actions, and defining SLAs and KPIs for the service.

This concept may be new to the world of service orchestration, but it’s standard procedure in traditional software development; whenever developers write code, it’s best practice to also develop test cases for that code as part of the process. Fortunately, the semantically rich YANG modeling language used to model services and devices in Cisco NSO is just as well suited to model assurance.

In a service orchestration context, operators can start at the highest level: What type of service will be delivered? Layer 3 VPN? Video conferencing? Voice over IP? From there, they can model the appropriate test actions for the service (activation testing, User Datagram Protocol [UDP] monitoring, video monitoring), and define the endpoint where they will be executed.

As part of this design-time process, operators also model the list of named SLAs for that service (for example, gold, silver, bronze), and configure SLA thresholds for that endpoint. Finally, they define the most relevant KPIs that will be measured directly at the customer endpoint to verify that the service actually experienced by the customer is meeting the SLA.

Network Services Orchestrator drives the instantiation of test agents from the NFV MANO to the endpoint. Those virtual probes will trigger the testing and monitoring actions from the assurance systems. And the data model can capture the output of those systems (activation test results, current KPI values, SLA status) as read-only values, which Network Services Orchestrator uses to orchestrate assurance at run-time.

Run-Time Assurance

Moving to run-time (the right-side circle in Figure 10), Network Services Orchestrator then configures the network according to the data model. First, it configures the physical devices and virtual functions for the service, similar to a conventional orchestration system. It also initiates activation testing as part of the same overall flow. And it initiates ongoing monitoring of service KPIs against SLAs—again, all based on the same YANG data model for the service.

As discussed, the best way to answer the core assurance question (Is the customer happy?) is to measure KPIs directly at the endpoint. Traditionally, this would require onsite equipment and expensive truck rolls. But since Network Services Orchestrator sees and orchestrates the entire network topology and endpoints, it can instantiate virtual test probes, or configure available measurement features like Cisco IP SLA, to measure those KPIs in all the right places.

Finally, for element 5 of the right-side wheel in Figure 10, Network Services Orchestrator orchestrates the ongoing monitoring and assurance functions. This includes automatically driving changes to assurance systems in response to any change in the network or service. If the orchestrator adds a new virtual component, adds or removes a leg to an existing VPN, changes a quality-of-service (QoS) configuration, or makes any other modification to a service, Network Services Orchestrator once again activates its state convergence algorithm. In this way, it can ensure both that those changes are fully tested at activation, and that ongoing assurance systems account for them.

Also tightly coupled as part of this process, Network Services Orchestrator updates traditional service impact systems. Now, any dynamic changes to the service or topology reflected in classical assurance tools—alarms, inventories, device monitors—also trigger Network Services Orchestrator to re-run its state convergence algorithm, creating a closed feedback loop.

The end result: Operators can break down the silos between fulfillment and assurance. They can reduce delivery times and support on-demand services. And they can automatically assure services against SLAs by directly measuring the most relevant KPIs, instead of hoping that an accurate picture of the customer experience filters up from the fog of device-level monitoring data in the network.
Cisco Network Services Orchestrator in Action: Orchestrating Service Fulfillment with Assurance

The following use case illustrates how Network Services Orchestrator can automate the full lifecycle of a network service. It details the design and instantiation of a Layer 3 VPN, focusing on the design-time and run-time assurance actions that can be orchestrated in an operator network. While this illustration provides a relatively high-level view of the processes involved, it is based on a real-world proof-of-concept (PoC) demonstration, using running code and the actual Network Services Orchestrator user interface.

Figure 11 shows an overview of this Layer 3 VPN scenario, encompassing three different demarcation devices:
- Low-end customer premises equipment (CPE) with no testing or operation, administration, and maintenance (OAM) support
- Thin CPE with hardware support for reflectors (such as Two-Way Active Measurement Protocol [TWAMP] and Y.1731)
- High-power compute CPE that's been deployed with VNF infrastructure

For the purposes of this illustration, assume that a virtualized WAN accelerator resides on the off-net CPE, and that several virtualized functions reside in the data center (virtual firewall, router, and Session Initiation Protocol [SIP] gateway). Note that this illustration includes off-net delivery—a scenario that can be error-prone, especially when QoS is provided over that path.
Also note that for all three of these endpoints, the operator will use direct measurement methods to assure the service. This will accomplish three things:

1. **It will enable automated activation testing at the time of initial delivery.** That includes carrier-grade network-level measurements as defined by the Metro Ethernet Forum (MEF). It also encompasses service validation across service chains, all the way from the CPE to the VNFs in the data center. This will allow Network Services Orchestrator to recognize, based on actual measurements of performance at the endpoint, whether it can move on to the ongoing assurance phase of the service lifecycle, or whether it needs to reconfigure or redeploy something that’s failed.

2. **It can enable SLA monitoring.** As specified in the service data model, Network Services Orchestrator will orchestrate monitoring of the relevant service- and customer-specific KPIs at the endpoint to gain real-time insight into the customer experience of the service. Network Services Orchestrator can extend this information to customer-facing portals, as well as for internal alarm and reporting systems. Most important, it can recognize, at run-time, when an SLA is violated and take action.

3. **It can enable Network Services Orchestrator to continue validating the customer experience across the service lifecycle.** Whenever a VNF in the service is upgraded, redeployed, or changed, Network Services Orchestrator will trigger service validation to verify that the change has not degraded the service from the customer’s perspective. And this happens automatically, at the speed of software.

All three of these direct measurement-enabled capabilities are essential for orchestrating service assurance synchronously with fulfillment, and, ultimately, automating service delivery end to end. Let’s take a closer look at how an operator would configure them.

### Designing Activation Testing Models

As an open platform, Network Services Orchestrator can integrate third-party assurance solutions as part of service orchestration. In this scenario, the operator is using the third-party Netrounds tool to configure in-depth activation testing and design virtual test probes that will be used to assure the service. Figure 12 shows the operator building a customized test sequence for the specific service being designed.

![Building Automated Activation Testing](image-url)
For this VPN service, the operator selects:

- RFC6349 TCP throughput test
- Service-related tests to validate that the SIP media gateway works and that voice quality is meeting customer expectations
- HTTP DNS testing
- IPTV testing

The operator also configures an activation test to validate the specific ports used in the service. And a QoS policy profile test is configured to validate that application traffic at the endpoint is being prioritized as expected. Note that this is all derived from the service data model in Network Services Orchestrator. The attributes that have been defined for the service can now be modeled in the testing domain.

This entire design test sequence can now be captured as a template, which Network Services Orchestrator uses as part of the orchestrated fulfillment process, within the same service data model.

Network Services Orchestrator provides service- and customer-specific parameters as input variables in the template, such as the defined SLA thresholds, which test agents should be used in the tests, and any other relevant parameters. It then initiates this testing sequence automatically when the service is activated, but also over the lifecycle of the service, whenever anything changes along the VPN path.

**Modeling Ongoing SLA Monitoring**

To assure the VPN against the customer’s SLA on an ongoing basis, the operator will take a similar approach. Once again, the operator uses the third-party Netrounds tool to configure continuous SLA monitoring, in this case this monitoring of UDP queues, HTTP, and some SIP endpoints (Figure 13).

As with the activation test sequence, these monitoring actions are captured in a template that Network Services Orchestrator uses to perform ongoing SLA assurance at run-time. And once again, this becomes part of the same service data model.

**Figure 13.** Defining SLA Assurance Monitoring
Instantiating Activation Testing and Assurance

Figure 14 provides a high-level view of how Network Services Orchestrator maps these orchestrated assurance functions to the ETSI NFV MANO framework to deliver the VPN service.

In this scenario, Network Services Orchestrator is acting as both the OSS and the NFV orchestrator, with Cisco Elastic Services Controller as the VNF manager and OpenStack for the virtual infrastructure manager (VIM). The operator uses Netrounds to provide a centralized test controller and virtual test agents. The VPN service will use a Cisco virtual cloud services router (Cloud Services Router 1000V Series) for the virtual premises equipment router, and a TelcoSystems CloudMetro compute CPE running on Intel hardware.

Figure 14. Mapping to ETSI NFV Reference Architecture

For the off-net endpoint with NFV infrastructure, the operator will use the full-power CloudMetro CPE to perform activation testing and SLA assurance monitoring. To address the low-end CPE with no testing capabilities, the operator adds an inexpensive physical x86 server running the same code. For the thin CPE endpoint with reflector support, the operator will not be able to perform the full range of activation testing and SLA monitoring as the other endpoints. But the assurance system will still be able to directly measure the service experience much more thoroughly than with standard connectivity tests.

Figure 15 shows the full range of testing and assurance functions that the operator can now bring to the service—all orchestrated by Network Services Orchestrator.
At both the low-end CPE and the off-net endpoint, the operator can now capture direct, granular metrics about the performance of the VPN service as the customer experiences it, including stateful TCP and advanced QoS tests. Even in the thin CPE, TWAMP and Y.1731 can provide real-time insight into important network performance metrics such as throughput, delay, and jitter.

As part of activation testing and ongoing service assurance, the operator can now also validate services across service chains (for example, measuring download rates, response times, TCP connect times, and more). And all of these testing and assurance processes are now automated synchronously with service fulfillment in Network Services Orchestrator, using the same data model.

**Orchestrating Service Assurance Functions**

Those are some of the in-depth assurance operations that the operator configures for the VPN service. Now, let’s see how the Network Services Orchestrator will orchestrate them as part of the VPN service (Figure 16).
First, based on the service data model, Network Services Orchestrator drives the device configuration to create the customer-specific VPN, transforming service data model parameters into device data model parameters automatically. Network Services Orchestrator configures the devices used in the VPN service, and instantiates and configures the virtual functions, including the virtual PE router and the virtual test agents.

As part of this process, Network Services Orchestrator uses MANO Network Service Descriptors (NSDs) to automate the deployment of those virtual probes to the correct place in the network. This is a major difference from classical test probes, which have to be manually relocated when the customer network changes. With Network Services Orchestrator state convergence, if the customer moves or adds an endpoint in the VPN service, all of the virtual probes and service descriptors move synchronously to the right place in the network automatically.

Once the virtual devices in the VPN service are up and running, Network Services Orchestrator drives the service-specific configurations.

And, as part of the same orchestrated transaction and data model, it triggers the assurance system to verify that both the VPN and the off-net service are delivering the expected performance.

This begins with the service-specific activation testing sequence that the operator has configured, which is now part of the data model. Once completed, Network Services Orchestrator hands off to the assurance system for ongoing SLA monitoring. Network Services Orchestrator provides the VPN topology, the endpoints, and the defined SLA parameters, so that the assurance system can now measure KPIs directly against the SLA for the customer’s service, on an ongoing basis.

**Service Orchestration with Assurance in the Network Services Orchestrator User Interface**

Figure 17 shows what this looks like for the operator using the Network Services Orchestrator UI. Here, Network Services Orchestrator displays the real-time topology of the VPN service, including three customer endpoints (CE devices), the virtual provider edge router, and virtual test agent.
Architecture for Lifecycle Service Automation:
Cisco Network Services Orchestrator Enabled by Tail-f
White Paper
Cisco Public

At the center-right, the square represents the NFV orchestrator responsible for instantiating the virtual services in the NFV MANO. Clicking it, the user can see that the Cisco virtual cloud services router and virtual test agent are booted and live, and that the test probe is in the correct place in the network for direct measurement.

On the right, the UI topology displays the leg in the VPN linking to customer CPE. By clicking that device, the user can see real-time assurance information, including SLA status and the history of activation tests that have been run on the endpoint.

The green circles show that there are no alarms for the device, and that it passed the activation test sequence when it was orchestrated. In this sample illustration, the red circle is indicating an SLA violation for that endpoint in the VPN service.

Hovering over the endpoint (as shown in the box on the far right), the operator can also view the current, service-specific run-time KPIs for both video and networking over the VPN. Note that Network Services Orchestrator is displaying a direct real-time measurement, not an inference based on data fetched at some time in the past.

Conclusion

The future of network services is rapidly approaching, and it’s no secret what the customer experience will look like. Operators will be able to design new services easily, abstracting away the complexities of physical and virtual infrastructure, arcane management systems, and diverse network domains. Those services will be tailored to each customer’s needs and preferences. They’ll be monitored at all times from the customer’s perspective to assure they’re delivering the experience customers expect. And they’ll be delivered in minutes, or even seconds, on demand.

By implementing the four pillars of successful lifecycle automation, Network Services Orchestrator can deliver the future of network services today. Drawing on common formal data models and data model mapping, it activates services programmatically, without manual coding. It employs state convergence to translate high-level service intent into network actions, and automatically respond to changes anywhere in the network.

It orchestrates services end to end across multiple domains. And it addresses the fully lifecycle of a service—fulfillment, activation testing, and ongoing monitoring and assurance—in a single transaction.

These are not planned capabilities that Network Services Orchestrator might enable one day in the future. They’re implemented in running code that operators can deploy today to become faster and more agile.

To learn more about Network Services Orchestrator, contact your Cisco account manager, or visit http://www.cisco.com/go/nso.