White paper
Cisco public

Single Source of Truth in Network Automation

Contents

This white paper discusses the importance of having a centralized and authoritative repository of accurate and current network information to make automation initiatives successful.

Introduction

In today's digital world, the **complexity of network environments is growing**. Networks have become increasingly complex, with a wide variety of devices, technologies, and configurations. Managing such complexity manually is no longer feasible, as it often leads to inefficiencies, inconsistencies, and human errors. **Automation has become a critical solution** to address these challenges.

Automation helps manage this complexity. Organizations automate network provisioning, configuration, troubleshooting and monitoring, aiming to streamline operations, to improve scalability, agility, and reliability, and to ensure consistency across their network infrastructure.

Before implementing automation, it is important for organizations to standardize network design and configurations. Without standardization, organizations risk creating unique and difficult-to-maintain configurations, which we refer as "snowflakes." Standardization includes design blueprints, configuration templates, and tooling, among other elements.

In addition, **for network automation to be truly effective, it is essential to have a reliable Source of Truth (SoT); an authoritative data source where we can reliably retrieve accurate and up-to-date network data**, including device inventory, configuration parameters, and other relevant data.

This Source of Truth, hereafter referred to as SoT, **serves as a single point of reference** for network and associated IT system attributes, enabling automation tools to make informed decisions and execute tasks accurately. These tasks may include network provisioning, network configuration, creation of test plans, execution of test plans, or updating the network documentation. **Without a SoT, automation efforts may suffer from inconsistencies, outdated information, and potential misconfigurations**, therefore decreasing the benefits that automation can bring to network environments.

This white paper aims to provide a comprehensive perspective on the importance of a SoT in network environments based on real-world experience working with diverse customers. The paper explores typical challenges faced by network operations teams in managing network complexity, and the relevance of a SoT to support the automation journey.

Through the study of **various use cases from production environments**, this paper offers insights into best practices and considerations for handling a SoT effectively. The intention of this publication is to assist organizations to make informed decisions on implementing and managing a SoT in their network environments.

It is also relevant to discuss the role of Artificial Intelligence (AI) in network automation. The AI revolution encourages business leaders to seek improved operational efficiency, which can result in increased returns or reduced costs. The ultimate goal of automation is "autonomous networks" with automated provisioning of self-organized, self-diagnosing, and dynamically updated networks. In autonomous networks, business intent and cross-domain insights—powered by AI systems—drive network changes to achieve outcomes that have a tangible and measurable impact on the lines of business, with IT teams taking an observer and supervisor function [1].

Artificial Intelligence for IT Operations (AIOps) is a term coined by Gartner in 2016 as an industry category for machine learning analytics technology that enhances IT operations analytics. Such operation tasks include automation, performance monitoring, and event correlations, among others. The goal is to enable IT transformation, receiving continuous network insights, and providing continuous fixes and improvements via automation [2].

Through structured data and automation, organizations can implement Artificial Intelligence solutions that build the foundation toward autonomous networks. By early adoption of the principles outlined in the white paper, organizations can transition faster towards stability, standardization, and autonomous networks.

Understanding Sources of Truth (SoT)

Organizations often rely on a wide range of Sources of Truth (SoT). These include shared Excel files, network documentation, change management (CMDBs) and classic databases, IP Address Management (IPAM) systems, Network Management Systems (NMS), Version Control Systems (VCS) such as Git, and network discovery tools among other options. Network devices such as routers or switches also have their own SoT, which is the state available on the device, often collected and updated via CLI commands. In the case of Software-Defined Networking solutions (SDN), network controllers aim to represent and manage the state of devices within a centralized SoT, which is often managed via a GUI or API.

In network automation, the primary function of a SoT is to provide an accurate, current, and reliable representation of network data being used by automation tooling. It acts as a single point of reference for all relevant provisioning, configuration, and testing details.

To achieve reliable and scalable network automation, a **best practice is to aggregate all the network and related system data, from their respective Sources of Truth (SoT) into a Single Source of Truth (SSoT)**. Centralizing and consolidating data into a single repository **facilitates automation systems in accessing the necessary information** more efficiently, thereby reducing the complexity and overhead of managing multiple data sources. This approach also eliminates data silos and ensures that automation tools access the most up-to-date information. This contributes significantly to streamlining network operations.

Network operations teams can accomplish critical network automation tasks by utilizing a SSoT, in configuration management, change management, compliance management, or monitoring and troubleshooting. The SSoT stores data such as corporate sites, buildings, network device hostnames, interfaces, IP addresses, VLANs, routing protocol attributes, network policies and more. Automation tools utilize this data to automate the provisioning and deployment of network components, as well as to generate and run test plans, and even to update network documentation.

In addition, automation tools can retrieve information from network devices and compare it with the data in the SSoT. The SSoT provides information about network topology, device relationships, and capacity metrics, so that automation tools can use this data for monitoring network health, detecting anomalies, and facilitating efficient troubleshooting.

Vendor-Specific SSoT vs. Open-Source SSoT

When it comes to adopting a SSoT tool, organizations need to decide between vendor-specific and open-source options. Each of them has advantages and disadvantages, and the decision needs to consider factors such as functionality, scalability, cost, and support. Both short-term and long-term strategy are relevant when making the decision.

Vendor-specific SSoT tools can result in easier integration when paired with automation tools from the same vendor. They typically come with technical support from the vendor. In many cases, they offer more advanced features. On the other hand, the cost is higher due to their licensing. Also, vendor-specific tools can impose limitations on organizations that aim for multi-vendor architectures that require integration of diverse vendor solutions.

Open-source SSoT tools provide better flexibility and lower cost. The support is provided by the community of users and through self-support. **The open-source tools facilitate the integration of multi-vendor network environments, facilitating the replacement of devices from different vendors without impacting the SSoT's functionality**.

A vendor-agnostic open-source SSoT future-proofs the automation initiatives, enabling organizations to evolve their networks according to technology trends, leveraging the best solutions available in the market, while retaining the SSoT. **In general, opting for a vendor-agnostic SSoT is considered best practice and offers several advantages**.

Standardization of data models

A **data model** is a **representation of the entities structure, relationships, and attributes** of the data within a system or domain. It provides a conceptual or logical framework for organizing and describing data elements and their interactions. In the case of network data models:

- **Entities** include sites, buildings, network devices, network device interfaces, virtual overlay networks, endpoints attached to the network, etc.
- **Relationships** define association of entities. For example, a switch belongs to a building, and multiple access points are attached to a switch. These relationships help to define the hierarchy of the network entities.
- **Attributes:** These are mainly proprieties or the network entities, including configuration parameters such as Virtual Routing and Forwarding (VRF) names, IP addresses, network policies, and more.

Data models help **ensure data consistency, integrity, and accuracy** by settings rules and constraints for data representation and relationships. In addition, structuring data according to a data model enables efficient querying and retrieval of information from a Source of Truth, as well as support for data manipulation operations. Data models also facilitate communication and understanding among stakeholders in the organization.

In the context of network automation, **a well-designed data model is crucial for the success of automation projects**. The example in Figure 1 describes the VRF configuration element for the Cisco Nexus-as-Code data model. The Nexus-as-Code data model serves as the foundation for Infrastructure as Code automation for network fabrics based on Nexus devices.

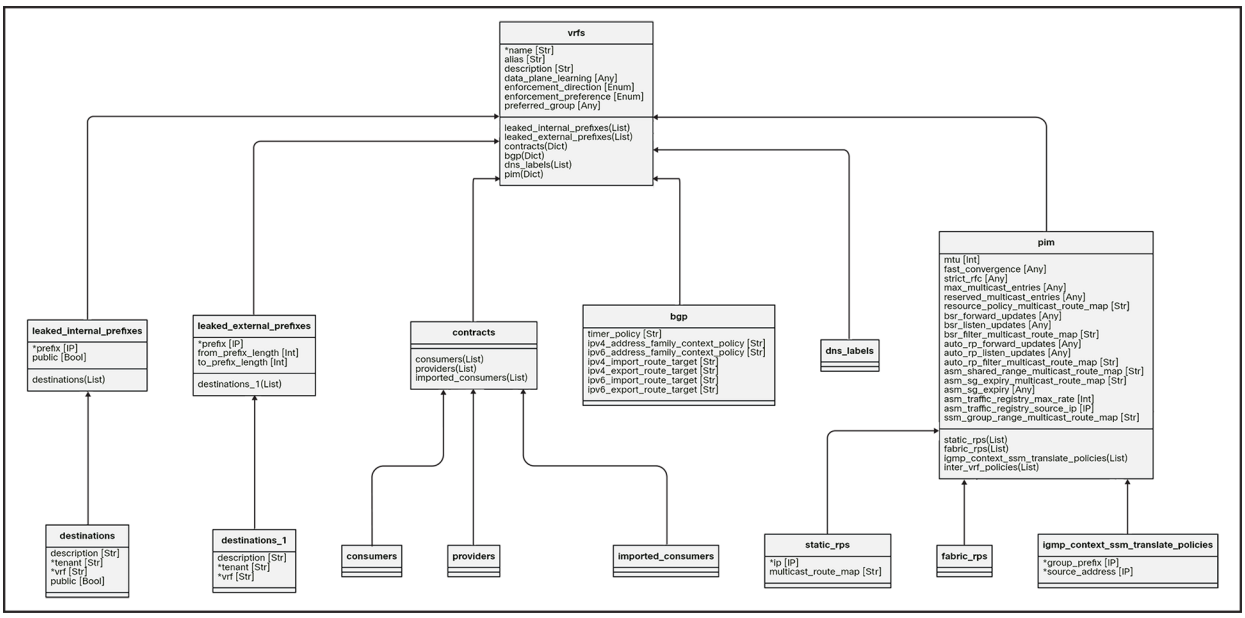

Figure 1. Data model for a network VRF in Cisco ACI® [3]

There is typically a **schema** definition associated with the data model, which is used to **validate the data input** files. A schema defines the entities' data types and the constraints that govern the data within that model; it outlines the rules for data validation of the data. Table 1 describes a section of the Cisco Nexus-as-Code data model's schema.

Table 1. A section of the Cisco Nexus-as-Code data model's schema

Name	Type	Constraint	Mandatory	Default Value
name	String	Regex: ^[a-zA-Z0-9_.:-]{1, 64}$	Yes	
alias	String	Regex: ^[a-zA-Z0-9_.:-]{1, 64}$	No	
description	String	Regex: ^[a-zA-Z0-9\\!#$%()*,-./:;@ _{\|}~?&+]{1, 128}$	No	
data_plane_learning	Boolean	true, false	No	true
enforcement_direction	Choice	ingress, egress	No	ingress
enforcement_preference	Choice	enforced, unenforced	No	enforced
bgp	Class	[bgp]	No	
dns_labels	List	String[Regex: ^[a-zA-Z0-9_.:-]{1, 64}$]	No	

When it comes to data models in the networking domain, there are two main categories: vendor-specific data models and vendor-agnostic data models. One well-known example of a vendor-agnostic data model is OpenConfig.

Vendor-specific data models are proprietary data models developed and maintained by networking equipment vendors; an example is the Nexus-as-Code model described above. Each vendor may have their own unique data model or set of models. These data models are typically designed to work specifically with the vendor's hardware and software platforms.

Vendor-agnostic data models, such as OpenConfig, are community-driven and collaborative efforts that aim to create standardized and vendor-neutral data models for network automation and management. OpenConfig defines and implements a common, vendor-independent software layer for managing network devices. OpenConfig is led by an Operator Working Group consisting of network operators from multiple segments of the industry [4].

Yet Another Markup Language (YAML) and JavaScript Object Notation (JSON) are popular file formats used for representing data models, offering a human-readable and machine-readable syntax. YAML is preferred for its readability and ease of use in configuration files, while JSON is widely used for data interchange between systems due to its simplicity and compatibility with multiple programming languages.

In the case of Cisco Nexus-as-Code, YAML format is used. When working with YAML files, the Yamale Schema Validator is a powerful tool that enables the definition of schemas using a YAML-based syntax. Yamale allows specifying rules and constraints for the structure and content of YAML data, checking whether the file adheres to the defined schema and providing validation results. Nexus-as-Code, mentioned earlier, utilizes Yamale for this purpose.

In contrast, YANG serves as a data modelling language primarily used in network management. It provides a standardized way to describe the structure, attributes, and relationships of data in network devices or systems. YANG models are commonly employed alongside the Network Configuration Protocol (NETCONF) or Representational State Transfer Configuration Protocol (RESTCONF) protocols to enable network management operations. Notably, OpenConfig data models are written in YANG.

A Single Source of Truth and an appropriate data model are the foundation for network automation, because they enable efficient data management and consistency.

The use of vendor-agnostic data models, such as OpenConfig, contributes to decoupling the management layer from vendor-specific implementations, helping organizations achieve interoperability and flexibility. However, it's important to acknowledge that vendor-agnostic data models have their own limitations, as there are often deviations or gaps in its coverage across network devices from different vendors. On the other hand, vendor-specific data models align with specific vendor features and represent vendor-specific functionalities.

In Cisco's experience, most enterprise organizations rely on vendor-specific data models for implementation of automation.

Application of multiple Sources of Truth

Sources of truth within a network can vary based upon the best and most valid datasets for the respective data types. In the context of network automation, different types of data may have their own authoritative sources that provide the most accurate and reliable information.

Examples of disaggregated Sources of Truth (SoT) include:

- Parameterized **network device configuration templates** that serve as standardized, reusable configurations that are populated—manually or by automation tools—with specific network configuration parameters and deployed to network devices. These templates act as a SoT by providing a standardized and controlled way of configuring network devices. The use of configuration templates helps enforce best practices, reduce configuration errors, and enable faster deployment of consistent configurations across the network infrastructure.
- **Configuration Management Databases (CMDB)**, a database that stores data in an IT environment. It provides a centralized repository for storing network device data attributes such as serial number, purchase date, installation location, and more. In the case of service provider networks, it also stores end user data such as subscriber contract, service subscription details, service level agreements, and other relevant customer related data. Automation tools use this data to configure network profile (e.g. QoS) within the network.
- **Network documentation** such as business and technical requirements, network functional requirements, architecture principles, solution design, network diagrams, device locations, or connectivity information also serve as SoT for network architecture. This network documentation provides operations teams with comprehensive and accurate details, aiding in understanding, troubleshooting, and effective management of the network infrastructure.
- **IP Address Management (IPAM)** systems that track and manage IP addresses, subnets, and DNS configurations. They maintain a database that stores information about available IP address ranges, allocated IP addresses, associated subnets, DNS entries, and other relevant IP addressing data. This database acts as the SoT for IP address–related information. IPAM plays an important role within any IT organization and has a heavy linkage to routing, switching, and security. Avoiding duplication of IP addressing in networks can help avoid costly outage scenarios and unnecessary operational overhead.
- **Network Management Systems (NMS)** that gather and consolidate data from various network devices and systems. They typically function as a centralized SoT for network monitoring. They provide comprehensive visibility, enable proactive management, and facilitate efficient troubleshooting and performance analysis of networks.
- **Network discovery tools** that scan the network and collect information about devices, including their IP addresses, interfaces, and relationships. They play an essential role in network management, serving as a SoT for operational data that is commonly used to validate other sources of truth.
- Vendors often offer a **network domain controller** for their environment (campus, data center, wide area network, service provider IP transport network etc.). They serve as centralized platforms where administrators define and manage network configurations, policies, and device settings. These components have an up-to-date view of the network from their perspective, and often that view could be considered the SoT for a given service context.

- In legacy networks, the **device configurations running on network devices** serve as the definitive and authoritative SoT. The running configuration represents the actual state of the device at a given moment and is crucial for maintaining accurate and consistent network operations. However, it is important to note that the running configuration may not always represent the desired state of the network. The desired state, to achieve optimal performance and functionally, is typically defined and maintained in a separate SoT.

As discussed earlier in this paper, a **best practice is to aggregate all the network and related system data, from their respective Sources of Truth (SoT) into a Single Source of Truth (SSoT)**. Centralizing and consolidating data into a single repository **facilitates automation systems in accessing the necessary information** more efficiently. Automation tools will need only to integrate with the SSoT, rather than with each individual and disaggregated SoT described above.

Earlier in this paper, we have discussed vendor-specific and open-source options. Through following the mantra of open-source SoT options vs. vendor-specific, the access to key and critical operational data and best practices in relation to network infrastructure and dependent components can be simplified. This data represents key artifacts that may be used for the enrichment of digital-twin operational modeling, and AI augmentation supporting AI Ops requirements, needed by organizations. This is why maintaining full control of data is relevant today and will remain relevant in the future.

Stateful Validation of Truths

When utilizing a SSoT within a deployment for configuration parameters, the SSoT becomes the central location for key attributes to be deployed within the network for a given data type. This however does not mean that configurations based on those parameters are blindly deployed within an environment without the required verifications and controls to handle conflict.

A simple example of conflict resolution in conjunction with a SSoT is for data related to Dynamic Host Configuration Protocol (DHCP). Most IP Address Management (IPAM) systems handle addressing allocations in a number of ways for address assignment. IP addressing pools, for instance the range 10.100.200.0/24, are reserved for usage by clients that map to a specific logical interface or virtual LAN. Within the configured pool, there may be static exclusions applied to ensure that already allocated addressing is not reassigned, such as the default gateway. Further to the initial exclusions within the IP range, there may also be the need to create static reservations for selected systems such as printers.

While it is possible to allocate a static client IP mapping directly on the DHCP server, the correct place to perform that mapping would be at the SSoT. In a situation where such an allocation takes place on the SSoT, conflict validations routines should be executed prior to provisioning tasks taking place to ensure that there are no mismatches between the derived sources of data. Such back checks for verification can be performed based on a timer every 24 hours or every hour, or be event based.

Separation of Concerns

Earlier in this paper, we emphasized the best practice of aggregating all network and related system data from their respective Sources of Truth (SoTs) into a Single Source of Truth (SSoT).

However, **organizations that implement large and complex automation projects recognize that having a monolithic SSoT may not always prove to be the most effective approach**. Instead, they find it beneficial to split functions and utilize different tools managing different types of data. This "separation of concerns" allows for greater flexibility, modularity, and ease of management in various aspects of large and complex network infrastructures.

When considering SSoT separation of concerns, a typical approach is to use a Source Code Management (SCM) tool to store **static data** such as network configuration templates, while using a relational database for **dynamic data** such as IP addresses.

For **dynamic data, the use of a relational database as SoT** provides numerous benefits for finding and managing data. It offers structured data storage and facilitates efficient data retrieval. The database enforces data integrity, accuracy, and consistency. The database querying capabilities enable quick and targeted data searches, while relationships facilitate data correlation. Indexing optimizes search performance for faster results. The scalability of the database allows it to handle growing network data. Additionally, the relational structure enables meaningful reporting and analysis, empowering network administrators to gain insights from the stored data.

On the other hand, **for managing static data, a typical choice as SoT is SCM tools**. Static data for network automation primarily refers to configuration templates, which include static parameters such as Maximum Transmission Unit (MTU) settings and Border Gateway Protocol (BGP) timers that don't change frequently. The use of a SCM tool allows network administrators to take advantage of its version control capabilities, collaborative features, and ability to roll back to previous version of configurations if needed. SCM tools ensure the consistency and integrity of static data, and they provide a history of changes made over time.

Many relational databases solutions exist in the market, vendor-specific and open-source. One **popular tool for managing dynamic data for network automation is NetBox**. With its user-friendly web-based interface, NetBox provides a centralized platform for efficiently managing and visualizing network resources such as IP address and VLANs.

In the case of SCM, **popular tools for managing network automation data are GitLab and GitHub**. Both platforms offer robust version control features, allowing to track changes, and capabilities to interact with network automation CI/CD pipelines.

In large and complex automation projects where separation of concerns is desired, leveraging a Source of Truth like NetBox for dynamic data and Git for static data—together with the adoption of a CI/CD pipeline tool deployment and testing—enables network administrators to build scalable network automation workflows.

Source of Truth in action: Use cases and examples

Relational database with Cisco NSO in Service Provider networks

Cisco Network Services Orchestrator (Cisco NSO) is a model-driven network orchestration platform that supports multi-vendor networks through a rich variety of Network Element Drivers. It is widely used by Cisco customers—mainly in the service provider vertical—that have implemented network orchestration to simplify the entire lifecycle management for network services.

In many ways, Cisco NSO already acts as a Source of Truth for customer's network. Within its Configuration Database (CDB), Cisco NSO keeps track of the intended network configuration state, and will work hard to enforce this state onto the running network. While the network operator wants Cisco NSO to perform this role, organizations use tools such as NetBox to document many aspects of the desired configuration state of the network. The inventory of network devices under Cisco NSO management is the first aspect that we may need to tackle for the integration of NetBox and Cisco NSO.

Both Cisco NSO and NetBox offer robust APIs and well-formatted data models, which can be leveraged to generate the device inventory for Cisco NSO from NetBox in an ad-hoc fashion. Integrations can also be achieved using built-in capabilities from either Cisco NSO or NetBox directly, rather than relying on an independent script or utility, often referred to as "middleware." An example of such integration is described in an article, "NetBox as Inventory Source of Truth for Cisco NSO," in the Cisco developer blog [5].

Practical examples of integrations between Cisco NSO and NetBox are also described in Cisco Live session DEVNET-2459, and code is available at https://github.com/annately/nso-netbox-cl. The first example includes the creation of a NetBox Form, defines new sites, selects the next-available IP prefix and reserves it, reserves DNS entries, creates devices, assigns the IP addresses to interface, and adds devices to Cisco NSO. The second example provides code that gets the information on site subnet from NetBox, reserves a new pool in NetBox for site DHCP server, and configures the router as DHCP server.

These are just examples on how service providers using Cisco NSO integrate with other SoT tools, such as NetBox.

In the context of Cisco NSO integration with NetBox, there are also examples of customers that are fetching all data from NetBox and storing it in Cisco NSO CDB, creating the concept of a "NetBox cache" within Cisco NSO, so that they can save the time required to call the NetBox API each and every time.

Relational database as a SoT in large and complex Enterprise Infra-as-Code project

In response to a customer in the financial industry facing scaling issues with their Infrastructure-as-Code (IaC) solution, Cisco collaborated with the customer to design and develop a solution for this challenge. It involved integrating NetBox and Git as SoT tools into their CI/CD pipeline.

The focus of the use case presented in this section is not on the motivation for the SoT separation of concerns, but on how to address a challenge frequently observed as the IaC journey progresses, becoming large and complex: how to manage the environment state, or truth, as the solution scales. This is often observed in two outcomes:

- **Vertical scaling** in a single technology domain, for example, when an infrastructure-definition YAML file becomes many thousands of lines long. Splitting large YAML files into smaller files by function can help but does not entirely resolve the problem.
- **Horizontal scaling** across multiple technology domains, for example, maintaining many YAML files across multiple Git repositories.

The vertical scaling challenge in this customer was due to data duplication in files. To illustrate this, consider a Cisco Application Centric Infrastructure (Cisco ACI) network where 1000 Gateways/Bridge Domains (GW/BD) need to be created, and each GW/BD definition needs 4 lines of code; that amounts for a total of 4000 lines. However, out of all that code only 25% is unique—the specific line with the IP address. Here, Cisco collaborated with the customer to develop reusable templates for ACI tenant configuration. As new tenants were requested by users through an API, an automated process would deploy the tenant, but render the template with dynamic assigned variables provided by NetBox relational database, using API GET calls for "Next Available IP Address" or "Next Available Prefix." In this case, we could reduce the length of the file by a factor of 4.

For the horizontal scaling challenge, a relational database ensures that customers can share information across technology domains, but simultaneously ensures there was a Source of Truth for their shared data. Using the same method of dynamic assignment, it helps avoid issues such as IP address/prefix conflicts, and instead allows the customer to validate the data is unique to a device.

Cisco Learning and Certification

Within the Cisco Learning and Certification organization, the team has taken an automation-first approach to deploying the network architecture end-to-end. When the Learning and Certification organization were confronted with the need to move their legacy data center from a location in Research Triangle Park in North Carolina, to Richardson in Texas, the technical team seized the opportunity to revamp the digital documentation, ensuring that all relevant information was thoroughly documented. This included:

- Rack layouts
- Inventory data
- Connection data
- Storage clusters
- Virtual Machines
- IPAM (DNS, DHCP)
- VLANs
- Interface descriptions (Logical/Physical)
- Power entry units
- Power distribution
- Secrets

For the items described above, after long evaluations, the team decided to use **multiple SoT**, using the right tool for the right job.

For inventory and IPAM-related parameters, including the ones described earlier, the decision was to go with the open source **NetBox** project. Given the rich community supporting the project, and its adoption in the field, it was selected to be the best choice for the task.

To manage secrets, passwords, certificates, and other security-related data, the use of **HashiCorp Vault** as a Source of Truth was decided upon.

Gitlab was used to further augment certain metadata which didn't necessarily fit directly into NetBox and stored in YAML format.

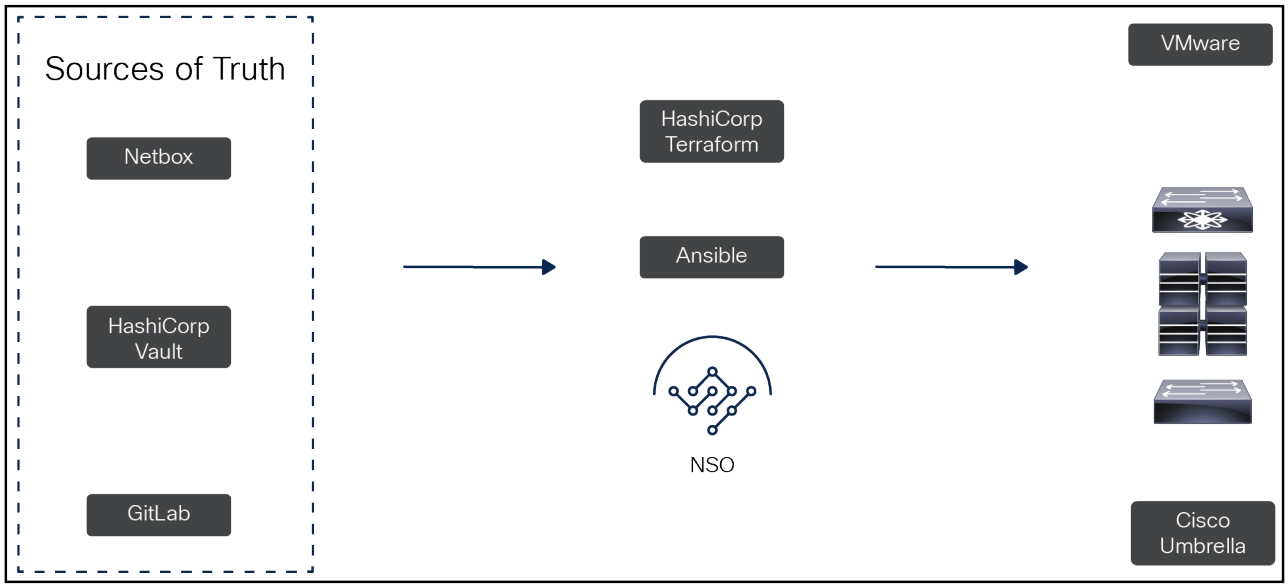

Figure 2. Learning and Certifications Data Center Services

Through maintaining the right level of operational rigor around populating and maintaining the Sources of Truth, consistency in how the network is operated, updated, and automated is vastly simplified. Further integrations into Cisco Modeling Labs testing infrastructure and Observability and Network Monitoring systems became possible, increasing the operational efficiency of the deployment.

Relational database as a read-through cache in European public sector

In the context of implementing a Single Source of Truth, NetBox can also serve as a read-through cache. This approach proves particularly useful when there is a need to centrally merge data from different systems. Typically, this involves merging the data from these various systems through an Extract, Transform, Load (ETL) process:

- The **extract** step loads the data from the data-carrying system.
- The **transform** step transforms this data by applying rules and functions.
- The transformed data is then **loaded** into the target system.

Cisco was engaged to assist a customer in the public sector who had a requirement to merge data from multiple systems.

To accomplish this, the NetBox data model was extended with the customer-specific model so that the transform process could correlate the data from different systems.

Since the NetBox database was implemented as a read-through cache, the cache was invalidated with each ETL process. This means that whenever new data was extracted from the source systems, transformed according to the required format, and loaded into the NetBox database, the cache would be cleared or reset. This ensured that the most up-to-date and accurate data was available in the cache, reflecting any changes made during the ETL process. As a result, users could rely on the NetBox database as a reliable and current source of information for their operations.

Git as Single Source of Truth for Cisco Network-as-Code

Cisco Network-as-Code is an automation framework that aims to lower the barrier of entry to enterprise Infrastructure-as-Code automation through simplification, abstraction, and curated examples. It offers customers a solution designed to standardize the approach to automating networks with Cisco CX support.

With its declarative Infrastructure-as-Code paradigm, operation teams focus on describing what the automation must accomplish, rather than describing how to accomplish it, i.e., describe the explicit steps to accomplish the target network state. In Network-as-Code automation framework, the network operator focuses on defining the end state of the network.

The central objective of this project is aimed at leveraging a data model and automation modules that abstract the complexity of the native Terraform Providers and low-level Ansible modules. There are Network-as-Code solutions for different network domains, with Nexus-as-Code [6] being currently the most widely adopted by customers.

In Cisco Network-as-Code, the Single Source of Truth (SSoT) for network automation are YAML files in a Git repository. The network configuration inventory in Git represents the desired state of the network configuration, for both static and dynamic data. The data in the this SSoT is used for network configuration, testing, and documentation tasks.

In order to simplify the management of SSoT data in Git repository:

- The framework separates data (variable definition) from logic (infrastructure declaration), where one can be updated independently from the other.
- The desired configuration definition in Git repository is split into multiple YAML files, each covering a specific logical section of the configuration. Examples of configuration sections include tenants, access_policies, fabric_policies, etc.
- Pre-defined default values based on common best practices.

An example of YAML file structure for Nexus-as-Code is illustrated in Figure 3.

```
$ tree -L 2
.
├── data
│   ├── apic.yaml
│   ├── access_policies.yaml
│   ├── fabric_policies.yaml
│   ├── node_policies.yaml
│   ├── pod_policies.yaml
│   ├── node_1001.yaml
│   ├── node_101.yaml
│   ├── node_102.yaml
│   ├── tenant_PROD.yaml
│   └── defaults.yaml
└── main.tf
```

Figure 3. Sample YAML file structure in Nexus-as-Code

The numerous successful production deployments of Cisco Network-as-Code solutions serve as great example of the use of Git as a Single Source of Truth (SSoT) for Infrastructure-as-Code automation.

Conclusions

In today's digital world, the complexity of network environments is growing. Networks have become increasingly complex, with a wide variety of devices, technologies, and configurations. Managing such complexity manually is no longer feasible, as it often leads to inefficiencies, inconsistencies, and human errors. Even operators taking a script-based approach to network management still have significant hurdles in the path to autonomy and AI. Data model-driven automation has become a critical solution to address these challenges and will continue to be a key foundation for future and modern networks.

For effective network automation, the use of reliable Sources of Truth is essential. These centralized repositories provide accurate and up-to-date network information, preventing inconsistencies, outdated data, lack of standardization, and potential misconfigurations.

To simplify network operations, it is a best practice to aggregate all relevant network automation data from multiple sources of truth into a Single Source of Truth (SSoT). This consolidation helps ensure consistency and accuracy. It also simplifies the integration with automation tools.

Vendor agnosticism is a key consideration when establishing a Source of Truth (SoT). By adopting vendor-agnostic open-source tooling, the SoT facilitates the integration of multi-vendor network environments and allows for the replacement of devices from different vendors without impacting the Source of Truth's functionality and more importantly, providing an unfettered means for customers to maintain control and access to their data.

Stateful validation of truths is another important aspect. While a Source of Truth serves as a mechanism for reconciling the intended state of the network, it is important to ensure that this intent is correctly validated. To prevent potential network outages caused by malicious or accidental user actions, proper change control and testing processes must be in place.

This white paper provides a comprehensive perspective on the significance of concepts pertaining to a Source of Truth (SoT) in network environments, drawing insights from real-world experiences with diverse customers. It includes actionable use cases and examples to inspire further exploration and understanding of the topic.

Authors

Josh Halley, Principal Architect, Cisco Customer Experience

Asier Arlegui, Principal Architect, Cisco Customer Experience

Matthew Howlin, Customer Delivery Architect, Cisco Customer Experience

Reviewers

Michael Wielpuetz, Customer Delivery Architect, Cisco Customer Experience

Robert Csapo, Product Manager, Cisco Engineering

Julio Gómez, Principal Architect, Cisco Sales

Michael Kaemper, CX EMEA CTO

Citations and Bibliography

[1] How automation is driving network engineer skills transformation, 2022.

[2] Wikipedia, Artificial Intelligence for IT Operations, 2023.

[3] Cisco DevNet, Nexus-as-Code, 2024.

[4] OpenConfig, 2024

[5] H. Preston, "NetBox as Inventory Source of Truth for Cisco NSO, 2021.

[6] https://developer.cisco.com/docs/nexus-as-code/, 2024.

Americas Headquarters
Cisco Systems, Inc.
San Jose, CA

Asia Pacific Headquarters
Cisco Systems (USA) Pte. Ltd.
Singapore

Europe Headquarters
Cisco Systems International BV Amsterdam,
The Netherlands

Cisco has more than 200 offices worldwide. Addresses, phone numbers, and fax numbers are listed on the Cisco Website at **https://www.cisco.com/go/offices**.

Printed in USA C11-4465315-00 08/24