Australia Telecom Service Operations Management

Last updated: 06/12/2026

Some examples and graphics depicted herein are provided for illustration only. No real association or connection to ServiceNow products or services is intended or should be inferred.

Please read the ServiceNow Website Terms of Use at www.servicenow.com/terms-of-use.html

Company Headquarters
2225 Lawson Lane
Santa Clara, CA 95054
United States
(408) 501-8550

servicenow

Table of Contents

Telecommunications Service Operations Management

Learn how Telecommunications Service Operations Management (TSOM) empowers communication service providers (CSPs) to proactively monitor, analyze, and resolve network and service issues before they impact customers. Built on the ServiceNow AI Platform, TSOM delivers a unified operations view across distributed, multi-domain telecom environments, helping teams improve service availability, operational efficiency, and customer satisfaction.

Get started

Explore

Learn about how telecom service providers use Telecommunications Service Operations Management.

Configure

Plan and configure your Telecommunications Service Operations Management.

Use

Use Telecommunications Service Operations Management to track comprehensive telecom service operations.

Reference

Get Telecommunications Service Operations Management reference information.

Additional resources

Some ServiceNow resources that can provide you with helpful information are:

Release Notes

Learn more about what's new and changed in this release at Telecommunications Service Operations Management (TSOM) release notes

ServiceNow Community

Connect with other Telecommunications Service Operations Management users at Now Community

ServiceNow Impact

Find useful resources related to your role and explore best practices at the ServiceNow Impact

ServiceNow University

Access real time courses, self-paced training, and career resources at ServiceNow University

Support

Contact customer service support at https://support.servicenow.com/now

TMF APIs for TMT

Review the multiple TM Forum (TMF) Open APIs supported for the product, see TMF APIs for TMT

Telecom data model

Explore the CI classes and relationships that represent telecom network elements in the CMDB at Telecom data model

Exploring Telecommunications Service Operations Management

Learn how Telecommunications Service Operations Management (TSOM) empowers communication service providers (CSPs) proactively monitor, analyze, and resolve network and service issues before they impact customers. Built on the ServiceNow AI Platform, TSOM delivers a unified operations view across distributed, multi-domain telecom environments, helping teams improve service availability, operational efficiency, and customer satisfaction.

Telecommunications Service Operations Management is a solution for telecom providers that offers complete visibility into network and service health. TSOM collects, correlates, and prioritizes events from across network domains—such as access, transport, and core—using standard APIs and the power of the ServiceNow AI Platform. By providing real-time, actionable insights, TSOM enables both frontline and back-office teams to address service-impacting issues and maintain consistent performance.

How TSOM works

Telecommunications Service Operations Management simplifies telecom operations by connecting with existing monitoring and telemetry platforms, identifying actionable patterns, and automating resolution workflows. It leverages:

- External event management via Telecommunications API notifications for standardized alarm ingestion.
- Event Management for event correlation, suppression, and prioritization.
- Metric Intelligence to detect threshold breaches, performance degradation, and anomalous behavior in real time.
- Health Log Analytics to analyze log data, triage related issues, and identify root causes before users are impacted.
- Service Impact Analysis to assess and trace service disruptions based on impacted infrastructure and business services.
- Service Graph Connectors and Discovery to build a dynamic, telecom-aware CMDB.

TSOM Architecture and Telecom Applications

Key features

TSOM key capabilities

Capability	Description
Real-time event monitoring	Ingest alarms and events from multi-domain network monitoring systems using External event management via Telecommunications API notifications.
Event correlation & analysis	Leverage Event Management and Metric Intelligence to correlate related events, reduce noise, and detect anomalies. For more information, see Event Notification Management Open API.
Telecom Visibility	Gain end-to-end visualization of network and service health, including impact traceability.
Service Impact Analysis	Understand how network or infrastructure issues affect services and prioritize remediation based on business impact.
Metric Intelligence	Monitor performance trends, detect threshold breaches and anomalies in metrics to proactively identify issues. For more information, see Metric Intelligence.
Health Log Analytics	The ServiceNow Health Log Analytics application helps prevent IT issues before your users are affected. It helps you identify the root cause of an issue by enabling you to triage related logs and analyze the raw data. For more information, see Health Log Analytics
Automated remediation	Use guided workflows and playbooks to drive fast, consistent, and auditable issue resolution.
Telecom-aware CMDB	Link infrastructure, services, and physical/logical configuration items (CIs) using a telecom-specific model for accurate root cause analysis.
Alert Management	Manage alerts efficiently with correlation, grouping, and automated response actions. For more information, see Manage and monitor alerts and Fault Management: Events and alerts.

Key benefits

- MTTD and MTTR reduction: Identify and respond to service-impacting issues before escalation.
- Improve operational efficiency: Noise reduction and streamline operations with guided workflows.
- Gain End-to-End Visibility: Understand how infrastructure issues impact customer-facing services.
- Integrate seamlessly: Ingest data from legacy and modern NMS/EMS tools with open standards.
- Ensure compliance: Align with TM Forum standards for telecom service assurance.

Key personas

- Network Operations Center (NOC) agents: Monitor network events and execute guided resolutions.
- Service Assurance managers: Analyze service trends and track resolution metrics.
- System Integrators / admins: Configure data ingestion, correlation rules, and workflows.
- Back-office analysts: Investigate root causes and track CI relationships in the CMDB.

Integration with the ServiceNow AI Platform

TSOM is embedded into the Now Platform and integrates seamlessly with core capabilities such as:

- Discovery
- Service Mapping
- CMDB
- Flow Designer and Playbooks

This integration confirms consistent workflows, accurate service models, and unified operations management across telecom domains.

Related topics

External event management via Telecommunications API notifications

Telecom Visibility

Telecom data model

The Telecom data model defines a structured framework for representing telecommunications networks within the CMDB. It extends the core Configuration Item (CI) model to describe telecom-specific infrastructure, connections, and relationships in a consistent and scalable way.

At its core, the Telecom data model organizes network elements such as equipment, interfaces, ports, sites, and connections, and defines how these elements relate to one another. The model captures both physical and logical aspects of the network and uses containment and consumption relationships to represent how network components are assembled and interconnected. This structure enables accurate modeling of complex telecom environments while remaining aligned with the standard CMDB architecture.

The Telecom data model is designed to be extensible and technology-agnostic. It provides a common foundation that supports multiple telecommunications domains by reusing existing CMDB classes where possible and introducing new classes only when telecom-specific representation is required.

Key components

- Network equipment and hardware: Physical and logical equipment, along with the components they contain, such as interface cards and ports.
- Interfaces and ports: Network interfaces and ports that enable connectivity between devices and services.

- Sites and locations: Logical and physical locations used to group and organize network infrastructure.
- Links, circuits, and connections: Physical and logical connections that describe how network elements communicate and exchange data.

Benefits

By providing a consistent way to model telecom infrastructure and relationships, the Telecom data model enables accurate network inventory, impact analysis, and operational insight. The shared structure supports integration across systems, improves data consistency, and simplifies analysis of network topology and dependencies. This common modeling approach helps teams manage complex networks more effectively and supports use cases such as service visibility, operational monitoring, and lifecycle management.

Domain-specific **data models**

Domain-specific data models build on the Telecom data model by refining and specializing its classes for particular network technologies. These models reuse the core structure of the Telecom data model while introducing domain-specific attributes and relationships as needed. The SD-WAN data model is one example of a domain-specific data model. It focuses on the subset of Telecom data model concepts that are relevant to SD-WAN environments and applies them to represent SD-WAN topology, services, and operations, while remaining consistent with the broader Telecom data model.

Related topics

SD-WAN data model

Data model

SD-WAN data model

The ServiceNow AI Platform® uses a custom data model that defines how SD-WAN connectors discover and retrieve device information.

The high-level Telecom Data Model, part of the Common Service Data Model (CSDM), does not define how specific technologies such as SD-WAN should be modeled. To address this gap, the SD-WAN data model provides a dedicated set of classes and relationships that define the structure of an SD-WAN network within the Configuration Management Database (CMDB).

The model documents comprehensive details about network assets, including configuration data, available ports, and bandwidth allocations across sites and services. This enables centralized management of SD-WAN repositories, streamlined provisioning and monitoring, and efficient resource allocation for building and maintaining network infrastructure.

The Discovery Builder framework supports the SD-WAN data model. You can use the framework to create and customize discovered objects that map directly to SD-WAN-specific classes and relationships, such as controllers, edge devices, tunnels, overlays, and underlays. Discovered data is mapped to the appropriate SD-WAN classes rather than to generic network CI types, supporting downstream assurance workflows such as fault management and performance monitoring."

ServiceNow AI Platform® SD-WAN data model architecture

The following table describes the classes used in the SD-WAN data model and the type of data each class stores.

SD-WAN data model classes

Class	Description
Group	Discovered organizations or administrative domains (ADOMs). Required
Company	Discovered organizations or administrative domains
Network site	Network site, including geographical information
Network service instance	Logical grouping of devices and their associated configurations
Network equipment models	Product model of the equipment
IP router or any other supported equipment classes	Hardware or network gear classification, used for storing information about network equipment
Key values	Additional attributes or fields related to a CI that aren't covered by predefined CI columns
Network interface	Network interface on a piece of equipment
Network adapter	Network adapter on a piece of equipment
IP address	Management IP address associated with a device
Per-device license	License assigned to a specific device. Optional
Location	Location of a network site

MPN data model

The Mobile Private Network (MPN) data model extends the Telecom data model to represent both the physical hardware and the virtual network functions that make up a mobile private network, along with the relationships that connect them.

A mobile private network contains a mix of physical and virtual objects. Physical objects include servers, firewalls, routers, switches, antennas, and radio units. Virtual objects include the network functions that run on those servers, such as the User Plane Function (UPF) and the Unified Data Management (UDM) function, along with their associated licenses. The MPN data model provides dedicated classes and relationships in the Configuration Management Database (CMDB) for these objects so that the physical and virtual components of an MPN can be modeled accurately and consistently. Accurate inventory and relationship data enables faster troubleshooting, supports impact analysis when faults occur, and provides the foundation for automated event and performance monitoring across MPN network elements.

The model spans two domains of the mobile network: the Radio Access Network (RAN), which connects user equipment to the network over the air interface, and the Mobile Core, which provides packet routing, session management, authentication, and policy control. The model also defines upper-layer logical objects that group RAN and Core components into end-to-end network service instances and assign them to network sites.

Physical hardware classes

Physical hardware classes represent the tangible equipment deployed in the RAN and the Mobile Core. The following table describes the physical classes used in the MPN data model.

MPN physical hardware classes

Domain	Class	Description
RAN	Remote Radio Unit	Radio frequency unit, typically mounted close to the antenna, that transmits and receives signals over the air interface.
RAN	Antenna	Antenna assembly used to radiate and receive signals to and from user equipment.
RAN	Baseband Unit	Equipment that processes baseband signals and connects radio units to the transport network.
RAN	Baseband Control and Interface Cards	Cards inside a Baseband Unit that provide control plane processing and interfaces to radio and transport networks.
RAN	Server	Compute platform that hosts virtualized 5G Distributed Units (DUs) and Centralized Units (CUs).
RAN	BBU chassis (`cmdb_ci_chassis`)	Physical enclosure that houses one or more Baseband Units and their interface cards.
RAN	Antenna Tower (`cmdb_ci_antenna_structure`)	Physical tower structure that supports one or more antennas and their associated radio units.
RAN	Indoor Radio Unit (IRU) (`cmdb_ci_indoor_radio_unit`)	Radio frequency unit deployed indoors to provide cellular coverage inside buildings.
RAN	Radio Dot/ Distributed IRU (`cmdb_ci_distributed_indoor_radio_unit`)	Distributed indoor radio unit that extends indoor coverage from a central Indoor Radio Unit.
RAN and Core	MPN box/ cabinet (`cmdb_ci_container_site`)	Outdoor or indoor cabinet that houses the BBU chassis and other RAN equipment at the cell site.
RAN	Cable (`cmdb_ci_cable`)	Physical cable connection between MPN hardware components.
Core	Firewall	Security appliance that enforces traffic policies at the boundaries of the Mobile Core network.
Core	IP Router	Layer 3 device that forwards IP traffic between network segments in the Mobile Core.
Core	IP Switch	Layer 2 device that forwards traffic between hosts and routers in the Mobile Core.
Core	Time sync source	Timing source that provides accurate clock synchronization to the mobile network.

MPN physical hardware classes (continued)

Domain	Class	Description
		Common sources include GPS receivers and other PTP grandmaster clocks.
Core	Server	Compute platform that hosts 4G and 5G virtualized network functions (VNFs).
RAN and Core	Network Adapter	Physical network interface hardware (cmdb_ci_network_adapter) installed in a Server, Router, Switch, or Firewall.
RAN and Core	Network Interface	Logical interface (cmdb_ci_network_interface) exposed by a Network Adapter and used to terminate IP connectivity.
RAN and Core	IP Address	IP address (cmdb_ci_ip_address) assigned to a Network Adapter or Network Interface and used for routing traffic across the MPN.

Virtual network function classes

Virtual network function classes represent the software components that run on the MPN servers. RAN virtual functions handle radio-side processing, while Core virtual functions provide subscriber, session, and policy services. The following table describes the virtual classes used in the MPN data model.

MPN virtual network function classes

Domain	Class	Description
RAN	Mobile Cell	Logical representation of a radio cell served by a Baseband Unit and its associated radio units.
RAN	vDU (virtual Distributed Unit)	5G virtual service that performs lower#layer baseband processing and connects to one or more radio units.
RAN	vCU-CP (virtual Centralized Unit, Control Plane)	5G virtual service (cmdb_ci_5g_cu_control_plane_network_function) that performs higher-layer control plane processing and signaling between the RAN and the Mobile Core.
RAN	vCU-UP (virtual Centralized Unit, User Plane)	5G virtual service (cmdb_ci_5g_cu_user_plane_network_function) that performs higher-layer user plane processing and forwards subscriber traffic to the Mobile Core.
Core	4G and 5G VNFs	Virtual services that represent Mobile Core network functions. The 5G Core includes nine functions: AMF (Access and Mobility Management), AUSF (Authentication Server), SMF (Session Management), UPF (User Plane), UDM (Unified Data Management), PCF (Policy Control), NRF (Network Repository),

MPN virtual network function classes (continued)

Domain	Class	Description
		NEF (Network Exposure), and NSSF (Network Slice Selection). The 4G EPC (Evolved Packet Core) includes six functions: MME (Mobility Management Entity), HSS (Home Subscriber Server), PCRF (Policy and Charging Rules), AAA (Authentication, Authorization, and Accounting), SGW (Serving Gateway), and PGW (Packet Data Network Gateway).
Core	VNF license	License assigned to a specific virtual network function. Tracks entitlement and consumption per VNF.
RAN	Virtual Machine	Compute instance (cmdb_ci_vm_object) that runs on a RAN Server and hosts a vDU or vCU virtual service.
Core	Virtual Machine	Compute instance (cmdb_ci_vm_object) that runs on a Core Server and hosts a 4G or 5G VNF.
Core	IMS network functions	Shared 3GPP network functions (cmdb_ci_network_function_instance) that provide IP Multimedia Subsystem (IMS) services for voice and multimedia sessions across 4G and 5G.
Core	MPN O&M functions	Operations and management applications (cmdb_ci_app) such as ProbeApp and Prometheus that monitor and manage the MPN.

MPN-specific classes

The MPN data model introduces custom classes that represent assets and resources unique to mobile private networks, including radio spectrum, network slices, data network names, and SIM provisioning. The following table describes the MPN-specific classes used in the MPN data model.

MPN-specific classes

Class	Description
Spectrum	Frequency band (u_mpn_spectrum) allocated to the mobile private network and consumed by Mobile Cells for radio transmission.
SIM range	Block of subscriber identities (u_mpn_sim_range) provisioned to the MPN for allocation to SIM cards.

Upper-layer **logical classes**

Upper-layer logical classes group the physical and virtual components of an MPN into end-to-end service constructs and assign them to a geographic site. The following table describes the upper-layer logical classes used in the MPN data model.

MPN upper-layer **classes**

Class	Description
Network service instance	End#to#end logical service that represents an entire MPN deployment, including its RAN and Mobile Core components.
Mobile Core network service instance	Logical service that groups the Mobile Core VNFs, supporting hardware, and licenses that deliver core packet, session, and policy functions.
RAN network service instance	Logical service that groups the RAN virtual functions, baseband and radio hardware, and Mobile Cells that deliver radio access.
Network Site	Physical or logical location to which the MPN network service instance and its components are assigned.
Slice	Logical network slice (u_mpn_slice) that partitions the MPN to deliver tailored connectivity for specific use cases or tenants.
DNN	Data Network Name (u_mpn_slice_dnn) that identifies the external data network reachable through a slice.
SIM card	Individual subscriber identity module (cmdb_ci_sim_card) assigned from a SIM range and consumed by user equipment connecting to a Mobile Cell.

Relationships between physical and virtual objects

The MPN data model uses CMDB relationships to express how physical equipment hosts virtual functions, how virtual functions consume each other, and how individual components roll up into network service instances and sites. These relationships make dependencies and connectivity visible in the CMDB and support downstream assurance workflows such as fault management, performance monitoring, and impact analysis.

- RAN servers Contain the Virtual Machines that host the vDU and vCU virtual services.
- Core servers Contain the Virtual Machines that host the 4G and 5G VNFs.
- Baseband Units Contain the Baseband Control and Interface Cards, Remote Radio Units, and Antennas.
- Mobile Cells are Provided By the Baseband Unit and its associated radio units.
- VNF licenses are Consumed By the VNFs they entitle.
- The MPN RAN network service instance and the MPN Mobile Core network service instance are Contained By the MPN network service instance.
- The MPN network service instance and its components are Located In an MPN Network Site.

Related topics

Configure elastic connectors for MPN alarm collection

Configure elastic event pull connectors for MPN

MPN Formula Engine processing

Metric-to-CI binding

Telecom Visibility

Telecom Visibility extends the discovery and reconciliation capabilities of ITOM Visibility to meet the specific needs of telecom service providers. It enables accurate, telecom-aware network inventory by discovering real-time network data, reconciling mismatches, and maintaining a consistent telecom model in your CMDB and TNI (Telecommunications Network Inventory).

Discover physical and logical network components across telecom domains, reconcile discrepancies between discovery and inventory data, and maintain a telecom-specific CMDB structure to support advanced automation and assurance use cases.

Telecom Visibility overview

Telecom Visibility is a unified solution for discovering and managing telecom network resources. It leverages proven discovery technologies such as Horizontal Discovery, Telecommunications Discovery Patterns, and Service Graph Connectors (SGC) to bring network data from various systems (for example, CLI/SNMP-based standalone devices or API-driven EMS/NMS/controllers) into your CMDB.

Telecom Visibility helps communication service providers (CSPs) and telecom operators:

- Discover and maintain telecom-specific network inventory
- Reconcile mismatches between planned/design inventory and actual network state
- Support Autonomous Network Operations (ANO), service fulfillment, and assurance workflows

Key Capabilities

Key capabilities of TSOM

Capability	Description
Telecom Discovery with Patterns	Discover standalone network functions (xNFs) such as routers and switches using SNMP and CLI protocols via predefined Telecom Discovery Patterns.
Service Graph Connector Integration	Bring inventory and configuration data from external EMS/NMS/ controllers into CMDB using northbound APIs.
Discrepancy Identification & Reconciliation	Detect and resolve mismatches between inventoried and discovered data (for example, missing CIs, model mismatches, stale entries).
Telecom-Centric Data Modeling	Maintain structured and hierarchical telecom data including interface cards, slots, subslots, VLANs, and ports.
CMDB Integration	Use CMDB 360 and Discovery Admin Workspace to monitor discovery sources, attribute changes, and reconciliation outcomes.

Applications and Plugins

1. Telecommunications Horizontal Discovery Patterns: Discovers telecom devices (xNFs) such as routers and switches using SNMP/CLI. Enables accurate inventory updates by scanning live network infrastructure. Supports vendor-specific patterns including:
   - Cisco (ASR1K, 7613, Nexus 9000, Nexus 3548)
   - Juniper (MX80, MX104, MX240, MX460)

2. Service Graph Connectors (SGC s) for TSOM: Prebuilt integrations with EMS/NMS/controllers to import inventory data via APIs. Supports:
   - Scheduled or on-demand discovery
   - Multi-instance configurations
   - Integration with systems like Nokia Altiplano (GPON networks)
3. Telecom Visibility Plugin: Provides shared logic and foundational components for both discovery and discrepancy reconciliation. Ensures:
   - Consistent behavior across TSOM workflows
   - Reusable components for reconciliation and remediation
4. CMDB CI Class Models for Telecom (Required version: 1.82 or later): Extends the CMDB to include telecom-specific classes and relationships, including:
   - Interface cards
   - Slots and subslots
   - Network interfaces

Telecom Visibility architecture

Telecom Visibility combines direct and indirect discovery methods:

- Direct Discovery using SNMP/CLI (for example, standalone routers/switches)
- Indirect Discovery using APIs from EMS/NMS/controllers (for example, Nokia Altiplano)

These feed into the CMDB and TNI, which then support decision-making, automation, and network operations. The following infographic demonstrates the architecture of Telecom Visibility.

Use Cases

Telecom Visibility use cases

Use Case	Outcome
Autonomous Network Operations	Reliable network inventory supports AI/ML-driven automation.
Service Fulfillment	Accurate CMDB/TNI ensures order accuracy and rapid provisioning.
Service Assurance	Reconciled inventory reduces troubleshooting time and increases uptime.
Audit & Compliance	Discovery Admin Workspace and CMDB 360 support audit readiness and traceability.

Key personas

Telecom Visibility is used by:

- Network and infrastructure teams: Discover and manage real-time network resource data across vendors and technologies.
- Inventory and fulfillment teams: Maintain an accurate and complete CMDB/TNI to support service fulfillment, order management, and provisioning.
- Platform administrators: Configure plugins, manage discovery sources, validate reconciliation rules, and monitor discovery performance.

Monitoring tools

Discovery Admin Workspace - Central console to:

- Monitor discovery tasks
- Validate reconciliation processes
- Tune schedules and view diagnostics

CMDB 360 - Visualize:

- Attribute-level update history
- Source attribution
- Reconciliation rules in action

Key benefits

Key benefits of Telecom Visibility

Feature	Benefit
CLI/SNMP Discovery	Expands network coverage without relying on external APIs.
Nokia Altiplano Integration	Supports GPON network discovery across multiple controller instances.
Discrepancy Reconciliation	Keeps inventory clean, actionable, and aligned with real network state.

Related topics

Configure Telecom Visibility

Discovery Admin Workspace

Telecom Visibility vs. ITOM Visibility

As telecom networks evolve and become more hybrid and complex, visibility into infrastructure is more critical than ever. To meet the unique needs of different environments, ServiceNow AI Platform offers two purpose-built visibility solutions: ITOM Visibility and Telecom Visibility.

While they're built on the same foundation, each is tailored to serve a different world—ITOM for traditional IT infrastructure and TSOM for telecom networks.

Both ITOM Visibility and Telecom Visibility are built on the same underlying capabilities of the ServiceNow Discovery engine, Identification and Reconciliation Engine (IRE), and CMDB. They both provide:

- Agent-based and agentless discovery.
- Horizontal and pattern-based discovery.

- Reconciliation of discovered data into CMDB.
- Dependency mapping and CI relationship creation.
- Integration with the Discovery Admin Workspace and CMDB 360.

Despite this shared architecture, the scope, use cases, and data models they serve are different.

Key differences between TSOM and ITOM Visibility

Telecom Visibility vs. ITOM Visibility

Feature / Focus Area	ITOM Visibility	Telecom Visibility
Target Environment	Traditional IT infrastructure (servers, applications, databases, cloud resources)	Telecom infrastructure (xNFs, network elements, EMS/NMS controllers)
Discovery Method	Horizontal Discovery with IT patterns	Horizontal Discovery with Telecommunications Discovery Patterns (SNMP/CLI) and Service Graph Connectors for telecom
CMDB Model	ITOM CMDB classes (for example, Windows Server, Application, Network Adapter)	Telecom-aware CMDB classes and Telecom Network Inventory (TNI) (for example, Interface Cards, Slots, LAGs, Subslots, VLANs)
Plugins required	com.snc.discovery and com.snc.itom.visibility	sn_tsom_core, sn_tsom_patterns, and telecom-specific SGC plugins (for example, sn_sgc_altiplano_connector)
Use Case Focus	Application dependency mapping, service modeling, cloud infrastructure discovery	Telecom network inventory discovery, reconciliation, autonomous network operations
Discrepancy Handling	General IRE reconciliation rules	Telecom-specific discrepancy identification and reconciliation (for example, hierarchy mismatches, attribute-level conflicts)
Vendor Data Ingestion	Primarily via discovery patterns	Strong emphasis on northbound API integrations using SGCs (EMS/NMS/controllers)
Network Types Supported	Enterprise networks, datacenters, cloud	Multi-vendor, multi-domain telecom networks (RAN, Core, Transport, Access)

IT Discovery vs Telecom Discovery

The following table helps you understand the differences between ITOM Visibility and TSOM Discovery.

ITOM Discovery	Telecom Discovery
Flat, simple or no hierarchy	Hierarchical, alignment to telecom models
Basic attributes	Advanced attributes
The network is the trusted source of truth	Inventory/CMDB design is the trusted source of truth
Simple CI identification and reconciliation	Complex telecom CI identification and reconciliation
Populates the CMDB with the CIs found in the network (everything found is written into the CMDB)	Validates that the network implementation is in sync with CMDB/TNI records as designed/ planned
Never miss a new CI for monitoring	Start monitoring new CIs after they have been validated

Key benefits

Use ITOM Visibility when:

- Discover IT infrastructure components (for example, Windows servers, cloud VMs, databases, load balancers).
- Primary goals include service mapping, operational resilience, or cloud optimization.
- Focus on ITSM, ITOM, or DevOps use cases.

Use Telecom Visibility when:

- Discovering telecom network infrastructure, including devices not managed by traditional IT systems.
- Dealing with telecom-specific network hierarchies like cards, ports, subslots, and LAGs.
- Relying on EMS/NMS/controllers as authoritative data sources.
- Need discrepancy detection and reconciliation tailored for telecom inventory models.
- Aligning with TM Forum standards, supporting autonomous network operations, or enabling closed-loop assurance.

Examples

Use Case	ITOM Visibility	Telecom Visibility
Discover a fleet of virtual machines in AWS	Yes	No
Ingest router and switch data from an EMS using APIs	No	Yes
Identify application-to-application dependencies	Yes	No
Detect and reconcile mismatches in telecom card hierarchies	No	Yes

While ITOM Visibility and Telecom Visibility both serve to populate and maintain an accurate CMDB, they are optimized for different domains. ITOM Visibility is geared toward enterprise IT environments, while Telecom Visibility is tailored for the specialized needs of telecom infrastructure discovery, discrepancy management, and inventory reconciliation.

Choosing the right visibility solution—or using both in tandem—confirms that you maintain trusted, domain-specific operational visibility across both IT and telecom landscapes.

Telecom Discovery

ServiceNow AI Platform Telecom Discovery helps you gain comprehensive visibility into your telecom network infrastructure by extending the capabilities of ITOM Visibility to support telecom-specific use cases.

Built for communication service providers (CSPs), this solution enables the discovery and mapping of network elements across multi-vendor environments using standardized protocols and integration with network management systems.

By combining Telecom Discovery plugins with the power of Service Graph Connectors and Discovery Patterns, you can automatically populate and maintain accurate records of your telecom resources in the Configuration Management Database (CMDB), providing a unified view of both IT and network infrastructure.

Note: Telecom Discovery is part of the TSOM Visibility subscription and aligns with the TM Forum Autonomous Network Operations (ANO) framework.

With Telecom Discovery, you can:

- Discover physical and logical telecom network resources across domains and vendors.
- Integrate with Element Management Systems (EMS), Network Management Systems (NMS), and SDN Controllers.
- Automatically populate and update CMDB/TNI records based on real-time network data.
- Discover standalone xNFs using SNMP and Command-Line Interface (CLI).
- Enrich CMDB data using Service Graph Connectors and specialized discovery patterns.
- Identify discrepancies between discovered network data and inventory records.
- Support automation use cases through consistent and accurate infrastructure visibility.

Integration with ITOM Visibility

Telecom Discovery is designed to complement existing ITOM Visibility features. You can:

- Leverage Horizontal Discovery and ITOM capabilities alongside TSOM plugins.
- Maintain consistent discovery practices for IT and telecom resources.
- Use the same CMDB data model to manage cross-domain service visibility.

This integration confirms unified asset management, faster issue resolution, and streamlined operations across both IT and network domains.

Customization with low-code/no-code tools

Provides intuitive design tools to extend discovery logic without writing code. You can:

- Build or modify custom Service Graph Connectors.
- Extend Telecommunications Discovery Patterns to match vendor-specific requirements.
- Accelerate onboarding of new device types and network domains.

This approach enables CSPs to stay agile and reduce time to value when expanding their discovery footprint.

Key components

- Telecommunication Discovery Patterns(sn_tsom_patterns): Provides patterns for SNMP-based discovery of standalone routers, switches, and xNFs. Includes Cisco and Juniper-specific discovery logic.
- Service Graph Connector for Nokia Altiplano(sn_sgc_altiplano_connector): Enables data collection from the Nokia Altiplano Access SDN Controller via REST APIs.
- Telecom Core(sn_tsom_core): Delivers foundational capabilities such as discrepancy identification, remediation logic, and shared telecom discovery features.

Related topics

Configure Telecom Visibility

Use Telecom Discovery patterns

Direct Discovery using Discovery Patterns

The Telecommunications Discovery Patterns plugin (also known as TSOM Patterns) extends ServiceNow AI Platform Telecom Discovery to support direct discovery of standalone network elements—such as routers and switches—without relying on traditional network management systems. These patterns enable Communication Service Providers (CSPs) to identify and map multivendor xNFs using protocols like SNMP and CLI.

Telecommunications Discovery Patterns provide a powerful, pattern-based approach to discover and manage telecom network resources directly from network elements. These patterns are especially valuable for discovering standalone xNFs (such as routers and switches) that aren't managed through traditional EMS/NMS systems.

This capability enhances visibility across telecom infrastructures and facilitates both vendor-neutral and vendor-specific device data is captured and reflected in the CMDB and Telecom Network Inventory (TNI), following the TNI data model.

Note: Telecommunications Discovery Patterns are part of the TSOM Visibility subscription and are available as a customer-visible plugin from the ServiceNow Store.

Key capabilities

- Direct Network Element Discovery
  - Use SNMP and CLI to communicate directly with physical network elements.
  - Discover physical network inventory such as interface cards, slots, ports, and devices without relying on EMS/NMS mediation.
- Automated CI Mapping and CMDB Integration
  - Discovered data is automatically mapped to telecom-aligned configuration item (CI) classes.
  - Integrates directly with the Identification and Reconciliation Engine (IRE) to promote accurate, non-duplicate CI records in the CMDB and Telecom Network Inventory (TNI).
- CMDB Accuracy and Compliance
  - Patterns trigger CMDB Compliance Certification Audits to detect mismatches or outdated records.
  - Supports discrepancy identification and remediation, helping maintain data integrity.
- Low-Code Pattern Customization: Extend or customize discovery patterns using low-code pattern designers to meet your specific network architecture and business needs.

How it works

Telecommunications Discovery Patterns are executed through Horizontal Discovery using the Nebula Discovery Language (NDL). A pattern is a sequence of steps that:

1. Establishes a connection to a target device.
2. Executes commands using SNMP, CLI, or both.
3. Extracts CI attributes and relationships.
4. Sends results to IRE for reconciliation and insertion into the CMDB and TNI.

The discovery logic follows the TNI data model, which restructures complex CI relationships (e.g., card-on-card scenarios) to align with telecom modeling best practices. For example, If a child card is discovered inside a parent card, Telecom Discovery synthesizes a subslot to insert the child, avoiding an invalid card-on-card configuration.

Architecture using Horizontal Discovery and Telecommunications Discovery Patterns

The following infographic is an example of the implementation for standalone SNMP or/and CLI xNFs.

Horizontal Discovery Application

The Horizontal Discovery application in ServiceNow is a versatile and highly scalable discovery engine to operate effectively across network, IT, and cloud environments, collecting data across multiple layers to provide a holistic view of the infrastructure.

For more information, see Horizontal discovery process flow with patterns.

Supported discovery patterns

A pattern is a sequence of commands to detect attributes of a configuration item (CI) and its outbound connections. Telecom Discovery provides a set of preconfigured Patterns that cover a wide range of network elements. ServiceNow offers several predefined TSOM patterns, including:

- Telecom Router Pattern – Generic SNMP-based router discovery.
- Telecom Cisco 7613 Router Pattern – For Cisco 7613 routers using SNMP.
- Telecom Juniper MX SSH Router Pattern – SNMP + CLI discovery of Juniper MX routers.
- Telecom Cisco Switch Pattern – SNMP-based Cisco switch discovery.
- Telecom Switch Pattern – SNMP-based generic switch discovery.

TNI entity creation

If your instance has the **Telecom Network Inventory (TNI)** plugin installed:

- Every discovered CI automatically results in a TNI entity record.
- The IRE payload includes mapping to both `cmdb_ci` and `tni_entity` tables.

This promotes seamless alignment between operational and inventory systems, which is essential for order fulfillment, assurance, and network planning.

MID Server

MID Server is a Java application that runs as a Windows service or UNIX daemon on a server within your local network. The ServiceNow® MID Server facilitates communication and data transfer between a ServiceNow instance and external applications, data sources, and services.

For more information, see MID Server.

Identification & Reconciliation Engine (IRE)

IRE offers a centralized framework for identifying and reconciling data from multiple sources. It verifies the integrity of the CMDB and some non-CMDB tables when various data sources are used to create or update CI records.

Related topics

Install Horizontal Discovery and set up Discovery Patterns

Telecom Router Pattern

The ServiceNow® Telecom Discovery application uses the Telecom Router discovery pattern to find SNMP-based routers in the network. Discovering some of these resources requires updating the Telecommunications Discovery Patterns (TSOM Patterns) from the ServiceNow® Store.

Telecom Discovery uses the Telecommunications Discovery Patterns to run Horizontal Discovery. This Telecommunications Discovery Pattern uses a set of SNMP requests to find, classify, and discover network elements.

Telecom Router pattern is part of the Telecommunications Discovery Patterns application (sn_tsom_patterns), which is part of TSOM Visibility.

Request apps on the Store

Visit the ServiceNow Store to view all the available apps, and for information about submitting requests to the store. For cumulative release notes information for all released apps, see the ServiceNow Store version history release notes.

Prerequisites

- Subscription to TSOM.
- Confirm that your network router devices have SNMP access.
- On the ServiceNow instance, configure SNMP credentials. For more information, see SNMP support for Discovery.
- For setting up TSOM Patterns, see Install Horizontal Discovery and set up Discovery Patterns.

Impacted CMDB CIs and CI Relationships (Physical Layer)

CIs	CI Relationships
IP Router CI	IP Router Device is represented by the IP Router CI. Table name: **cmdb_ci_ip_router** IP Router CI contains Slots or Network Interfaces.
Slot CI	Slots are represented by the Slot CI. Table name: **cmdb_ci_container_slot** Slot is contained by the IP Router. Slot CI contains the Interface Card.
Subslot CI	Subslots are represented by the Subslot CI. Table name: **cmdb_ci_container_subslot** Subslot is contained by Interface Card CI. Subslot CI contains the Interface Card CI.
Interface Card CI	Different types of cards are represented by the Interface Card CI. Fan and Power Supply Units are also represented by the Card CI. Table name: **cmdb_ci_interface_card** Interface Cards are contained by Slots or Subslots. Interface Cards can contain Network Interface or Subslots.
Network Interface CI	Any type of Network Interface is represented by the Network Interface CI. Table name: **cmdb_ci_ni_interface** Network Interface is contained by Interface Card, IP Router.

Telecom Cisco 7613 Router Pattern

The ServiceNow® Telecom Discovery application uses the Telecom Cisco 7613 Router discovery pattern to find SNMP-based Cisco 7613 in the network. Discovering some of these resources

requires updating the Telecommunications Discovery Patterns (TSOM Patterns) from the ServiceNow® Store.

Telecom Discovery uses the Telecommunications Discovery Patterns to run Horizontal Discovery. This Telecommunications Discovery Pattern uses a set of SNMP requests to find, classify, and discover network elements.

Telecom Cisco 7613 Router Pattern is part of the Telecommunications Discovery Patterns application (sn_tsom_patterns), which is part of TSOM Visibility.

Request apps on the Store

Visit the ServiceNow Store to view all the available apps, and for information about submitting requests to the store. For cumulative release notes information for all released apps, see the ServiceNow Store version history release notes.

Prerequisites

- Subscription to TSOM.
- Verify that your network router devices have SNMP access.
- On the ServiceNow instance, configure SNMP credentials. For more information, see SNMP support for Discovery.
- For setting up TSOM Patterns, see Install Horizontal Discovery and set up Discovery Patterns.

Impacted CMDB CIs and CI Relationships (Physical Layer)

CIs	CI Relationships
IP Router CI	IP Router Device is represented by the IP Router CI. Table name: **cmdb_ci_ip_router** IP Router CI contains Slots or Network Interfaces.
Slot CI	Slots are represented by the Slot CI. Table name: **cmdb_ci_container_slot** Slot is contained by the IP Router. Slot CI contains the Interface Card.
Subslot CI	Subslots are represented by the Subslot CI. Table name: **cmdb_ci_container_subslot** Subslot is contained by Interface Card CI. Subslot CI contains the Interface Card CI.

CIs	CI Relationships
Interface Card CI	Different types of cards are represented by the Interface Card CI. Fan and Power Supply Units are also represented by the Card CI. Table name: **cmdb_ci_interface_card** Interface Cards are contained by Slots or Subslots. Interface Cards can contain Network Interface or Subslots.
Network Interface CI	Any type of Network Interface is represented by the Network Interface CI. Table name: **cmdb_ci_ni_interface** Network Interface is contained by Interface Card, IP Router.

Telecom Juniper MX SSH Router Pattern

The ServiceNow® Telecom Discovery application uses the Telecom Juniper MX SSHRouter discovery pattern to find SNMP and CLI -based Juniper MX Series routers in the network. Discovering some of these resources requires updating the Telecommunications Discovery Patterns (TSOM Patterns) from the ServiceNow® store.

Telecom Discovery uses the Telecommunications Discovery Patterns to run Horizontal Discovery. This Telecommunications Discovery Pattern uses a set of SNMP requests to find and classify CLI over SSH to discover network elements.

Telecom Juniper MX SSH Router pattern is part of the Telecommunications Discovery Patterns application (sn_tsom_patterns), which is part of TSOM Visibility.

Request apps on the Store

Visit the ServiceNow Store to view all the available apps, and for information about submitting requests to the store. For cumulative release notes information for all released apps, see the ServiceNow Store version history release notes.

Prerequisites

- Subscription to TSOM.
- Verify that your network router devices have SNMP access.
- On the ServiceNow instance, configure SNMP credentials. For more information, see SNMP support for Discovery.
- For setting up TSOM Patterns, see Install Horizontal Discovery and set up Discovery Patterns.

Impacted CMDB CIs and CI Relationships (Physical Layer)

CIs	CI Relationships
IP Router CI	IP Router Device is represented by the IP Router CI. Table name: **cmdb_ci_ip_router** IP Router CI contains Slots or Network Interfaces.
Slot CI	Slots are represented by the Slot CI. Table name: **cmdb_ci_container_slot** Slot is contained by the IP Router. Slot CI contains the Interface Card.
Subslot CI	Subslots are represented by the Subslot CI. Table name: **cmdb_ci_container_subslot** Subslot is contained by Interface Card CI. Subslot CI contains the Interface Card CI.
Interface Card CI	Different types of cards are represented by the Interface Card CI. Fan and Power Supply Units are also represented by the Card CI. Table name: **cmdb_ci_interface_card** Interface Cards are contained by Slots or Subslots. Interface Cards can contain Network Interface or Subslots.
Network Interface CI	Any type of Network Interface is represented by the Network Interface CI. Table name: **cmdb_ci_ni_interface** Network Interface is contained by Interface Card, IP Router.

Telecom Cisco Switch Pattern

The ServiceNow® Telecom Discovery application uses the Telecom Cisco Switch discovery pattern to find SNMP-based Cisco switches in the network. Discovering some of these resources

requires updating the Telecommunications Discovery Patterns (TSOM Patterns) from the ServiceNow® Store.

Telecom Discovery uses the Telecommunications Discovery Patterns to run Horizontal Discovery. This Telecommunications Discovery Pattern uses a set of SNMP requests to find, classify, and discover network elements.

Telecom Cisco Switch pattern is part of the Telecommunications Discovery Patterns application (sn_tsom_patterns), which is part of TSOM Visibility.

Request apps on the Store

Visit the ServiceNow Store to view all the available apps, and for information about submitting requests to the store. For cumulative release notes information for all released apps, see the ServiceNow Store version history release notes.

Prerequisites

- Subscription to TSOM.
- Confirm that your network router devices have SNMP access.
- On the ServiceNow® instance, configure SNMP credentials. For more information, see SNMP support for Discovery.
- For setting up TSOM Patterns, see Install Horizontal Discovery and set up Discovery Patterns.

Impacted CMDB CIs and CI Relationships (Physical Layer)

CIs	CI Relationships
IP Switch CI	IP Switch Device is represented by the IP Switch CI. Table name: **cmdb_ci_ip_switch** IP Switch CI contain Slots or Network Interfaces.
Slot CI	Slots are represented by the Slot CI. Table name: **cmdb_ci_container_slot** Slot is contained by the IP Switch. Slot CI contains the Interface Card.
Subslot CI	Subslots are represented by the Subslot CI. Table name: **cmdb_ci_container_subslot** Subslot is contained by Interface Card CI. Subslot CI contains the Interface Card CI.

CIs	CI Relationships
Interface Card CI	Different types of cards are represented by the Interface Card CI. Fan and Power Supply Units are also represented by the Card CI. Table name: **cmdb_ci_interface_card** Interface Cards are contained by Slots or Subslots. Interface Cards can contain Network Interface or Subslots.
Network Interface CI	Any type of Network Interface is represented by the Network Interface CI. Table name: **cmdb_ci_ni_interface** Network Interface is contained by Interface Card, IP Switch.

Telecom Switch Pattern

The ServiceNow® Telecom Discovery application uses the Telecom Switch discovery pattern to find SNMP-based Telecom switches in the network. Discovering some of these resources requires updating the Telecommunications Discovery Patterns (TSOM Patterns) from the ServiceNow® Store.

Telecom Discovery uses the Telecommunications Discovery Patterns to run Horizontal Discovery. This Telecommunications Discovery Pattern uses a set of SNMP requests to find, classify, and discover network elements.

Telecom Switch pattern is part of the Telecommunications Discovery Patterns application (sn_tsom_patterns), which is part of TSOM Visibility.

Request apps on the Store

Visit the ServiceNow Store to view all the available apps, and for information about submitting requests to the store. For cumulative release notes information for all released apps, see the ServiceNow Store version history release notes.

Prerequisites

- Subscription to TSOM.
- Confirm that your network router devices have SNMP access.
- On the ServiceNow® instance, configure SNMP credentials. For more information, see SNMP support for Discovery.
- For setting up TSOM Patterns, see Install Horizontal Discovery and set up Discovery Patterns.

Impacted CMDB CIs and CI Relationships (Physical Layer)

CIs	CI Relationships
IP Switch CI	IP Switch Device is represented by the IP Switch CI. Table name: **cmdb_ci_ip_switch** IP Switch CI contain Slots or Network Interfaces.
Slot CI	Slots are represented by the Slot CI. Table name: **cmdb_ci_container_slot** Slot is contained by the IP Switch. Slot CI contains the Interface Card.
Subslot CI	Subslots are represented by the Subslot CI. Table name: **cmdb_ci_container_subslot** Subslot is contained by Interface Card CI. Subslot CI contains the Interface Card CI.
Interface Card CI	Different types of cards are represented by the Interface Card CI. Fan and Power Supply Units are also represented by the Card CI. Table name: **cmdb_ci_interface_card** Interface Cards are contained by Slots or Subslots. Interface Cards can contain Network Interface or Subslots.
Network Interface CI	Any type of Network Interface is represented by the Network Interface CI. Table name: **cmdb_ci_ni_interface** Network Interface is contained by Interface Card, IP Switch.

Indirect Discovery using Service Graph Connectors

ServiceNow Telecom Discovery using Service Graph Connectors (SGC s) enables you to seamlessly integrate network infrastructure data from external management systems, such as EMS, NMS, and SDN Controllers, into the Configuration Management Database (CMDB).

This approach helps Communication Service Providers (CSPs) maintain a current and accurate view of their multivendor telecom network resources, services, and configurations.

By leveraging predefined connectors and robust data transformation tools, you can unify your inventory across domains and ensure that the CMDB and Telecom Network Inventory (TNI) reflect real-time network insights aligned with your telecom data model.

Note: Telecom Service Graph Connectors are part of the TSOM Visibility subscription and extend the capabilities of the standard Service Graph Connector framework.

Service Graph Connector overview

With Telecom SGC, you can:

- Ingest data from EMS/NMS/Controllers using northbound REST APIs.
- Automatically populate and update CMDB and TNI records with enriched, telecom-aligned data.
- Reconcile incoming data with existing CI records using the Identification and Reconciliation Engine (IRE).
- Generate TNI entity records automatically when the TNI plugin is installed.
- Support discrepancy detection and remediation as part of Telecom Discrepancy Identification & Reconciliation.
- Use low-code tools to configure, test, and manage your integrations end-to-end.

Architecture overview

Telecom Service Graph Connectors rely on a modular and scalable architecture:

Key components of Service Graph Connector

Component	Role
Service Graph Connector	Defines the integration logic to extract and stage data from EMS/NMS systems (for example, Nokia Altiplano or Nokia NSP).
MID Server	Acts as a secure bridge between your ServiceNow instance and the external network system. For more information, see IntegrationHub ETL.
IntegrationHub ETL (3.2)	Provides a guided UI for creating, testing, and managing ETL transform maps. For more information, see MID Server.
Robust Transform Engine (RTE)	Transforms staged source data into CMDB-compliant records using defined ETL logic. For more information, see Create a robust import set transformer.
Identification and Reconciliation Engine (IRE)	Ensures data consistency and prevents duplicates by identifying and reconciling CIs.

Key components of Service Graph Connector (continued)

Component	Role
	For more information, see the CMDB Identification and Reconciliation (IRE).
CMDB/TNI	Stores structured, accurate telecom infrastructure data for visibility and downstream processes.

Supported Service Graph Connectors

- Nokia Altiplano SGC (sn_sgc_altiplano_connector): Integrates with the Nokia Altiplano Access Network SDN Controller via REST APIs.
- Cisco Meraki SGC (sn_sgc_meraki_connector): Integrates with the Cisco Meraki Dashboard via REST APIs.
- Fortinet SGC (sn_sgc_fortinet_connector): Integrates with FortiManager via REST APIs.
- Arista VeloCloud SGC (sn_sgc_velocloud_connector): Integrates with VeloCloud Orchestrator via REST APIs.

Note: The connector can coexist with IT and Cloud Service Graph Connectors (e.g., for servers, monitoring tools, IoT, etc.).

Key benefits

- Fast time to value – Use predefined, supported connectors that require minimal configuration.
- Multivendor support – Integrate with various management platforms across access, core, and transport networks.
- Model-aligned visibility – Ensure telecom-specific hierarchy and relationships are accurately modeled in the CMDB.
- Discrepancy detection ready – Feed network data directly into Telecom Discrepancy Identification & Reconciliation for CMDB compliance.
- Scalable integration – Leverage ServiceNow's proven integration framework built for performance and extensibility.

Telecom Network Inventory (TNI) data model

Telecom SGCs include logic to ensure compatibility with the Telecom Network Inventory (TNI) data model:

- When the TNI plugin is installed, each discovered network element automatically includes a TNI entity record.
- A `tni_entity` is created alongside its corresponding `cmdb_ci` record, using system-generated payload mappings (e.g., `inventory_category`).
- This ensures consistency across operational and planning systems.

If TNI is installed, a payload like the following one will be added to the IRE payload for each item (with inventory_category populated based on the className):

```
related = [{
            "className": "tni_entity",
            "values": {
                "inventory_category": ""
            }
        }];
```

As a result, the discovered CI is in both the cmdb_ci and tni_entity tables.

Related topics

- Telecom Discovery via Nokia Altiplano
- Telecom Discovery via Cisco Meraki SD-WAN
- Telecom Discovery via Fortinet SD-WAN

Telecom Discovery via Nokia Altiplano

The Service Graph Connector for Nokia Altiplano offers a telecom-aware integration that brings real-time network inventory from the Nokia Altiplano Access Network SDN Controller into your ServiceNow CMDB.

Designed for service providers and telecom enterprises, this connector enables complete visibility, control, and synchronization of your physical and logical network infrastructure. This integration uses REST APIs and a MID Server to deliver a telecom-model-aligned view of your network, enabling more accurate service modeling, inventory management, and operational efficiency.

Key benefits

- Accurate inventory synchronization: Automatically populate the CMDB with up-to-date physical and logical inventory from Nokia Altiplano using secure, REST API-based ingestion. This includes OLTs, ONUs/ONTs, interface cards, ports, slots, and logical connections.
- Telecom-Aware CI modeling: Model your network infrastructure in a telecom-aligned format using purpose-built CI classes and relationships. The connector ensures the accurate representation of devices and their dependencies—mirroring your actual network topology.
- Simplified and Guided Setup: Reduce time-to-value with a built-in guided setup that walks you through connection configuration, data source management, and import scheduling with ease.
- Flexible discovery options: Choose how and when to run discovery based on your needs —whether it's full bulk loads, targeted filtered discovery, or phased OLT-only imports. Apply custom filters by device IP or name.
- Multi-Instance support: Scale effortlessly by onboarding multiple Altiplano instances independently. Configure connection aliases and define import schedules per instance for complete operational flexibility.

Note: Supported Nokia Altiplano Controller minimum version is 24.6. For a general overview of Service Graph Connector technology, see Getting started with Service Graph Connectors.

Nokia Altiplano SGC Architecture

The following table illustrates the architecture of Nokia Altiplano service graph connector.

Nokia Altiplano 2.0 Service Graph Connector architecture

#	Stage	Location	Components or sub-flow	Description
1	Source	Network/ IT	Altiplano Controller (REST)	External system holding OLT and ONU data. REST is the active protocol for this connector.
2	Altiplano data source (custom script)	MID Server	• REST action (get access token) • REST action (get list of OLT devices) • REST sub-flow (get OLT and its related ONUs, physical and logical details) • Bulk data source • Filtering data source • Filter devices list	REST actions run on the MID Server to get the access token from Altiplano, retrieve the list of devices for discovery, and get full device details (OLT, ONU, physical, and logical).
3	Import sets	ServiceNow instance	Altiplano import set	Imports data from external sources, triggered by Scheduled Data Import. The bulk data source pulls all Altiplano data; the filtering data source pulls only data for OLTs that match the filter criteria. Use Connection selects the Altiplano connection alias. The import set includes: • OLT physical and logical • ONU physical and logical • Logical ports relationships • Logical connections between OLTs and ONUs
4	IH-ETL RTE	ServiceNow instance	• Clean, transform, enrich data • Map to CMDB classes and add relationships • Prepare payloads for IRE	Determines which classes are populated based on discovered data and models. Handles transform maps, references of models and BW, CI relationships, and CI references.

Nokia Altiplano 2.0 Service Graph Connector architecture (continued)

#	Stage	Location	Components or sub-flow	Description
5	IRE	ServiceNow instance	• IRE rules execution • CMDB integration	RTE output is automatically passed to the IRE as a payload. The IRE matches to existing CIs based on identification rules and creates new CIs if no match is found.
6	CMDB compliance	ServiceNow instance	• Audit • Follow-on tasks • Remediation (writes to CMDB and TNI)	Detects CMDB discrepancies, such as relationship mismatches or CIs present in the CMDB but not discovered. Creates a discrepancy report. Resolves discrepancies automatically or manually through the remediation workflow.

Use cases

The following are examples on how you can use the Nokia Altiplano Service Graph Connector:

- Automatically ingest and structure network data into ServiceNow's CMDB for both physical and logical network elements: This creates a telecom-model-aligned CMDB view that mirrors your live network environment.
  - Physical Components: OLTs, ONUs/ONTs, slots, subslots, cards, and ports
  - Logical Components: Logical ports, VLANs, Link Aggregation Groups (LAGs), and logical paths between devices
- Maintain the integrity of your network data with scheduled, automated reconciliation that:
  - Detects and responds to real-time changes
  - Prevents data drift or stale records
  - Supports operational processes like service assurance, order fulfillment, and network planning
- Discover beyond physical infrastructure by capturing and managing logical connectivity within ServiceNow:
  - Identify and model logical ports, LAGs, and logical paths like PON and VLAN connections
  - Link logical CIs with physical components using parent-child and member-of relationships
  - Enhance diagnostics, impact analysis, and service modeling by visualizing end-to-end logical topologies
- Leverage ServiceNow's built-in Extract, Transform, Load (ETL) framework to simplify and accelerate integration:
  - Predefined transformation maps and CI class definitions reduce development overhead
  - Reuse existing CMDB structures to minimize customization
  - Rapidly onboard new Altiplano instances and scale across your network infrastructure

Key capabilities and components

Key capabilities and components

Capability	Description	Supporting component
Telecom-aware CMDB modeling	Maps Altiplano physical and logical inventory (OLT, ONU/ONT, ports, slots, cards, interfaces) into CMDB with telecom-specific CI classes and relationships.	RTE, IRE, CMDB tables (`cmdb_ci_optical_line_terminal`, `cmdb_ci_optical_network_terminal`, etc.)
Automated data ingestion	Retrieves device inventory from Nokia Altiplano via REST APIs with secure, scheduled imports.	MID Server, Data Sources (SGC-Nokia Altiplano Bulk/ Filtered Discovery)
Custom discovery control	Choose OLT-only or include ONU data, apply filters by IP or name, schedule jobs per instance.	Import Schedules, System Properties (`sn_sgc_altiplano.enable_onu_disc` etc.)
Multi-instance support	Configure and manage discovery independently for multiple Altiplano controllers.	Connection Aliases, Credential Aliases
Parallel data processing	Improve performance by running concurrent data-source jobs for large datasets.	System Property: `sn_sgc_altiplano.parallel_number` Enable Parallel Loading
Guided configuration	Simplifies setup with step-by-step interface for creating connections, credentials, and jobs.	Guided Setup UI (Navigation: All > Service Graph Connectors > Nokia Altiplano > Setup)
Model-driven CI classification	Matches discovered devices to models and assigns correct CI class (OLT, ONU, ONT), or falls back to Network Gear.	Model Tables, System Property: `sn_sgc_altiplano.onu_ci_class`
Relationship	Establishes telecom-specific CI relationships (e.g., member-of, contains, logical path).	IRE, Logical Connections CI (`cmdb_ci_ni_logical_path`)
Dashboards and monitoring	View status, results, and errors of each run; filter by connector or timeframe.	Integration Commons for CMDB Dashboard
Scalable and reusable architecture	Decoupled data sources, transformation, and CI reconciliation for easy scaling and customization.	Import Sets (`sn_sgc_altiplano_tsom_inventory`) Transformation Maps, System Properties

CMDB Integrations Dashboard

The Integration Commons for CMDB store app provides a dashboard with a central view of the status, processing results, and processing errors of all installed Service Graph Connectors. You can see metrics for all integration runs. You can filter the view to a specific integration, a specific

time duration, or a specific integration run. For more details about monitoring integrations in the CMDB Integrations Dashboard, see Integration Commons for CMDB.

Related topics

- Configure Nokia Altiplano service graph connector
- System components installed with Nokia Altiplano

Telecom Discovery via Cisco Meraki SD-WAN

The Service Graph Connector (SGC) for Cisco Meraki provides a cloud-based management platform that provides a visual representation of network traffic flow between services and applications, enabling centralized configuration, monitoring, and management. Network administrators can easily discover and map services, enforce security policies, and troubleshoot issues in real time.

The Cisco Meraki SGC provides visibility, control, and synchronization of physical and logical network infrastructure for service providers and telecom enterprises. It leverages REST APIs and a MID Server to deliver a unified, telecom-model-aligned view.

Key benefits

- Automated Device Discovery: Periodically, as determined by import schedules, scans the Cisco Meraki dashboard to import all Configuration Item (CI) data.
- Real-time synchronization: Ensures the Configuration Management Database (CMDB) is up to date with the latest CI information.
- Visualization: Provides a graphical representation of Cisco Meraki networks, device relationships, and dependencies through the Integration Hub ETL.

Note: For a general overview of Service Graph Connector technology, see Getting started with Service Graph Connectors.

Cisco Meraki SGC Architecture

The Cisco Meraki architecture consists of the following key components:

- Setup: A guided multi-step process that establishes connection alliances, configures credentials, and sets up REST API endpoints to integrate with the Meraki dashboard.
- Data sources: Scripts that fetch information related to various components such as organizations, networks, devices, and uplinks from Meraki APIs.
- Import schedules: Determines when data sources are executed to facilitate the discovery and updating of CIs.
- Connection and credential aliases: Details of the protocols used for API requests to extract data.
- Connections: Contains the necessary connection details for accessing the Meraki environment during the discovery process.
- Credentials: Provides the necessary authorization information to obtain data from the Meraki environment.
- Properties: System properties that contribute to configuring Cisco Meraki SGC and can be referenced within data sources.

CMDB Integrations Dashboard

The Integration Commons for CMDB application provides a dashboard with a central view of the status, processing results, and processing errors of all installed Service Graph Connectors.

The dashboard displays metrics for all integration runs. You can filter the view to a specific integration, time duration, or integration run. For more details about monitoring integrations in the CMDB Integrations Dashboard, see Integration Commons for CMDB.

Related topics

- Configure Cisco Meraki Service Graph Connector
- Run and verify an import schedule for Cisco Meraki SGC

Telecom Discovery via Fortinet SD-WAN

The Service Graph Connector for Fortinet provides a telecom-aware integration that imports real-time network inventory data from the Fortinet network management system into the Configuration Management Database (CMDB). This integration enhances visibility and management of network resources.

Key benefits

- Accurate inventory synchronization: Automatically populate the CMDB with up-to-date physical and logical inventory from Fortinet using secure, REST API-based ingestion. SD-WAN devices, interface cards, ports, slots, and logical connections are included.
- Telecom-aware CI modeling: Model your network infrastructure in a telecom-aligned format using purpose-built CI classes and relationships. The connector promotes an accurate representation of devices and their dependencies that mirrors your network topology.
- Simplified and Guided setup: Reduce time-to-value with a built-in guided setup that walks you through connection configuration, data source management, and import scheduling with ease.
- Flexible discovery options: Choose how and when to run discovery. Apply custom filters by device IP or name.

Note: For a general overview of Service Graph Connector technology, see Getting started with Service Graph Connectors.

Fortinet SD-WAN architecture

The Fortinet SD-WAN architecture is composed of three key components:

1. FortiPortal: A cloud-based service through which you register for Fortinet accounts and manage their licenses.
2. FortiManager: A centralized management solution used to oversee and manage SD-WAN networks.
3. FortiAnalyzer: A security analytics and reporting tool that collects and analyzes logs from Fortinet devices to identify potential security threats and conformance issues. Alerts are sent through a preconfigured webhook, and metrics are retrieved through API calls.

Fortinet APIs use JSON-RPC to enable programmatic interaction with FortiManager and FortiAnalyzer for bulk configuration, device management, and inventory collection.

The following infographic illustrates the architecture of the Fortinet Service Graph Connector.

CMDB Integrations Dashboard

The Integration Commons for CMDB application provides a dashboard with a central view of the status, processing results, and processing errors of all installed Service Graph Connectors. The dashboard displays metrics for all integration runs. You can filter the view to a specific integration, time duration, or integration run. For more details about monitoring integrations in the CMDB Integrations Dashboard, see Integration Commons for CMDB.

Related topics

- Configure a Fortinet SD-WAN Service Graph Connector
- Run and verify an import schedule for Fortinet SGC

Telecom Discovery Builder framework

The Telecom Discovery Builder framework ETL (Extract, Transform, Load) is a reusable and schema-aligned component delivered with the Telecommunications Service Operations Management (TSOM) Core application. It provides a consistent and extensible method for ingesting telecom inventory data into the ServiceNow configuration management database (CMDB) across multiple Service Graph Connectors (SGCs).

The Telecom Discovery Builder framework ETL serves as a baseline data ingestion utility to handle telecom-specific configuration item (CI) data. It enables connector development teams to avoid building ETLs from scratch by offering a standardized transform logic that can be duplicated and customized for each connector. It provides a standardized, reusable foundation that promotes consistency across implementations.

Once the TSOM Core plugin is activated, the generic ETL is auto-provisioned and becomes available in the Integration Hub ETL Studio, ready to be reused and adapted to connector-specific needs.

Why and when to use the Telecom Discovery Builder framework

Use the Telecom Discovery Builder framework when you want to do the following:

- Deploy a new Service Graph Connector in a telecom environment and want to avoid building the ETL from scratch.
- Maintain schema-aligned consistency across multiple connectors and platform instances.
- Discovery payloads and CI relationships compliant with TNI (Telecommunications Network Inventory). For more information, see Telecommunications Network Inventory.
- Work within an SGC (Service Graph Connector) application scope and want to customize ETL behavior without modifying core logic.
- Benefit from predefined mappings, validated JSON schema support, and a UI-driven configuration interface.

Key capabilities and features

- Auto-provisioned with TSOM core: Installs automatically and is ready to use across telecom connectors.
- Generic schema-based ingestion: Supports a unified data schema for telecom CIs.
- Supports duplication and customization: Duplicate the Telecom Discovery Builder ETL framework into your application scope using Integration Hub ETL Studio.
- TNI support: Aligned to the TNI data model and the ETL logic can be extended to generate and link TNI entities.
- Flexible field-mapping interface: Configure import sets, data sources, targets, and transformation logic through a UI-driven experience.

Key user roles

The tsom_visibility_admin and tsom_assurance_admin roles provide granular access control across TSOM Visibility and Assurance application scopes, supporting stronger security and compliance in line with the Granular admin directive.

- The tsom_assurance_admin role provides full operational and administrative access to TSOM Assurance application scopes.
- The tsom_visibility_admin role manages the operational tasks for the TSOM visibility application.

These roles control access to the TSOM Visibility and Assurance application scopes listed in the following table.

TSOM Visibility and Assurance application scopes

Scope name	Description
`sn_tsom_core` (Telecom Service Operations Core)	Includes the audit application and various system properties within that app scope.
`sn_tsom_em_core`(Event Management Core)	Includes the monitoring and analysis of telecommunications network alerts and performance metrics from SD-WAN infrastructure.
`sn_tsom_em_connect`(Event Management Connectors)	Include push connectors and connector definitions used to ingest alerts from external telecommunications network sources.
`sn_tsom_patterns`(Telecom Discovery Patterns)	Include the MID Server and system IP Service properties.
`sn_sgc_aliplano`(Service Graph Connector for Nokia Altiplano)	Includes the application properties.
`sn_sgc_meraki`(Service Graph Conenctor for Cisco Meraki Telco SD-WAN)	Includes the application properties.
`sn_sgc_fortinet`(Service Graph Connector for Fortinet Telco SD-WAN)	Includes the application properties.

The following table lists the roles contained in the tsom_visibility_admin role.

Visibility granular admin roles

Role name	Description
`import_admin`	• Monitors scheduled data import executions, including viewing the hierarchy and flow of chained imports, accessing execution contexts, and diagnosing import issues. • Manages all aspects of import set records and imports.
`connection_admin`	• Responsible for managing connections and credentials, especially for Integration Hub. • Create and configure HTTP and HTTPS connections, which are used by custom actions or activities to connect to external endpoints. These actions include setting

Visibility granular admin roles (continued)

Role name	Description
	up connection details such as credentials, connection aliases, URLs, MID Server options, and additional attributes.
`sn_cmdb_admin`	• Highest-level administrative role for CMDB privileges. • Grants full access to CMDB data, tools, and user interfaces, and enables administrators to configure policies such as class management and application service requirements that aren't available to editors.
`discovery_admin`	• Grants access to the tables in the Discovery application. • Essential for configuring and executing discovery in your network. • Includes setting up MID Servers, managing credentials, defining discovery schedules, and validating results.
`usage_admin`	• For instances using Discovery, this role enables users to track the number of active users and hardware configuration items (CIs) discovered. • Monitors instance usage through dashboards, providing information to help your organization understand application adoption and usage patterns.
`cmdb_inst_admin`	• Enables users to create and customize integrations using Integration Hub ETL, including integrating third-party data into CMDB or supported non-CMDB tables. • Necessary to use the Integration Hub ETL store app and access CMDB Integrations.
`certification_admin`	• Enables users to manage certification filters for auditing subsets of records. • Create, update, delete, and run certification audits, compare records against expected values, and assign follow-on tasks to remediate discrepancies.

Visibility granular admin roles (continued)

Role name	Description
	• Manage and reassign follow-on tasks from compliance audits and view all such tasks. Users with the certification role can access only tasks assigned to them.
`agent_admin`	• Responsible for downloading and administering the system's built-in MID Server agent. • Enables users to manage scripts and properties related to the MID Server.

Key benefits

Benefits of the Telecom Discovery Builder framework ETL include:

- Consistency in how telecom configuration items (CIs) are structured and loaded into the CMDB.
- Reusability through duplication and customization across connectors.
- Compliance with TNI schemas and discovery model requirements.
- Flexibility to extend and tailor data transformations without compromising the base schema.

Related topics

Configure the Telecom Discovery Builder ETL

Extend TNI entity support for duplicated ETLs

Telecom Discrepancy Identification and Reconciliation

Use the Telecom Discrepancy Identification and Reconciliation capability to keep your CMDB or Telecommunications Network Inventory (TNI) in sync with your live telecom network. By continuously auditing and comparing discovered data with inventory records, this solution helps you detect, classify, and automatically remediate inconsistencies before they impact service quality, assurance, or fulfillment processes.

Telecom Discrepancy Identification and Reconciliation is a telecom-specific capability included with the TSOM Visibility plugin. It helps confirm the integrity of your network inventory by identifying mismatches between real-time network data (from Discovery or external systems) and inventory records stored in the CMDB or TNI.

This solution uses:

- CMDB Compliance Certification audits to detect anomalies.
- Follow-on tasks to log and track issues.
- Automated remediation subflows to resolve discrepancies.

Note: Keeping inventory accurate enables automation, reduces service errors, and supports regulatory compliance. It is also a foundational component of TM Forum's Autonomous Network Operations (ANO) framework.

Telecom Discrepancy Identification & Reconciliation architecture

Key features

Key features of Telecom Discrepancy Identification and Reconciliation

Feature	Description
Continuous audit checks	Compares discovered network data with inventory records using Certification Audits.
Discrepancy detection	Identifies CIs that are missing, misconfigured, or misaligned in terms of hierarchy or attribute values.
Follow-On task generation	Automatically creates tasks for each discrepancy to enable tracking and remediation.
Automated remediation	Uses Flow Designer subflows to resolve issues such as outdated CIs, invalid relationships, and missing discovery data.

How Telecom Discrepancy Identification & Reconciliation works

Once the discovery process is complete, the CMDB Compliance framework identifies mismatches between discovered data and inventory records using scheduled or on-demand compliance audits.

- Compliance Audits: Compare discovered network data with CMDB inventory records to detect inconsistencies.
- Follow-On Tasks: Automatically generated for each failed audit. These tasks document and categorize the identified discrepancies.
- Remediation Subflows: Launched from Follow-On Tasks to resolve discrepancies by updating, decommissioning, or realigning Configuration Items (CIs).

Note: For more information, see Discrepancy identification – types of discrepancies.

CMDB Compliance and Telecom Discrepancy Identification & Reconciliation

CMDB Compliance is a toolset that enables administrators to certify CMDB data for accuracy and identify discrepancies detected during compliance audits. It can also automatically generate and assign Follow-on Tasks for failed audit records, which serve as tasks to trigger an appropriate remediation subflow to correct discrepancies. CMDB Compliance Audits form the foundation of the Telecom Discrepancy Identification & Reconciliation.

- CMDB Compliance runs audits as a post-processing rule, identifying anomalies (discrepancies) in the CMDB.
- CMDB Compliance creates a Follow-On Task for each Audit Record in a failed state (the failed state is the result of an audit finding an anomaly or discrepancy in the CMDB). A remediation flow can be designed and triggered for each Follow-On Task to address and resolve the discrepancy.

The logic for Telecom Discrepancy Identification & Reconciliation, as well as the example remediation subflows, are automatically with the TSOM Visibility plugin. For more information on the general CMDB Compliance toolset, see CMDB Compliance.

Discrepancy Identification Scenarios (using Compliance Audits)

Discrepancy identification in TSOM Visibility relies on using CMDB Compliance (Certification Audits) and has extended it by adding specific logic that uses model relationships and information to identify mismatches. To support remediation, the system generates specific reconciliation task types for each issue found, such as:

- Slots occupied discrepancy
- Most recent discovery date not set
- Most recent discovery date not within configured threshold
- Model relationships not defined
- CI model not found
- Incorrect number of relationships
- Reference to Logical Interface not found

You can use the following audits to identify the discrepancies in the discovered physical and logical entities

- Telecom Discrepancy Audit
- Telecom Logical Connections Discrepancy Audit
- Telecom Network Topology Discrepancy Audit

Note: For more information on the general compliance audits, see Certification audits.

Automation and UI Integration

- Every failed audit automatically creates a Follow-On Task.
- Tasks can trigger prebuilt or custom subflows using Flow Designer.
- Manual remediation can be initiated using a "Remediate" UI action button in the task form.
- Remediation steps are recorded in work notes for visibility and auditing.

Note: For more information on how to build a subflow, see Building subflows.

Follow-On Task types created for failed Audit Result Records

The following discrepancy types (Audit Results) can be found for Parent CI and child CIs for each relationship record in the CI Relationship table (cmdb_rel_ci) that matches the conditions, and the following Follow-On Tasks can be created for each of the failed Audit Results:

1. The most recent discovery date not set- generated in case the Most recent discovery date field in CI is missing.
2. The most recent discovery date not within configured threshold- generated in case the difference in the Most recent discovery date field value between a Parent CI and child CI is more than 2.5 days. For example, By default, it is set to 2.5 days in the sn_tsom_core.discovered_date.diff.threshold.in.days system property and can be changed.
3. CI model not found–(the 'Model ID' field isn't set or data is invalid). Generated in case a corresponding CI model isn't found. If a CI model isn't found, the next validations (4-6) are irrelevant because they rely on CI models. In case a CI model is found, the audit will continue to the next validations (4-6).
4. Slots occupied discrepancy-Generated in case a card occupies an incorrect number of Slots.

5. Model relationships not defined-relevant only if TNI is installed. Generated if the audit is unable to find a relationship between Parent and child CI models in the Network Model Relationships table.
6. Incorrect number of relationships - relevant only if TNI is installed. Generated if the audit finds that the number of discovered child CI records exceeds the maximum number of its corresponding Parent CI record in the model relationship Count field in the Network Model Relationship table.
7. Incorrect number of relationships - generated during the Logical Connection Discrepancy Audit when a logical interface is associated with more than one logical connection, violating the expected one-to-one mapping.
8. Incorrect number of relationships - generated during the Network Topology Discrepancy Audit when a network topology record does not meet the required relationship criteria. Specifically:
   - The record must have at least one "Contains:Contained By" relationship with an equipment CI.
   - The record must also have at least one "Members:Member Of" relationship with a logical connection CI.
9. Reference to Logical Interface not found - generated during the Logical Connection Discrepancy Audit when a logical connection is missing one of the end points (Port A or Port z, or both).

Telecom Reconciliation

Automate reconciliation of network inventory discrepancies and enhance operational efficiency. Telecom Reconciliation helps to:

- Identify and resolve discrepancies between live network inventory and CMDB inventory to ensure alignment and boost productivity
- Empower users with auto generated discrepancy reports detailing the type of discrepancy
- Provide users with recommended corrective actions and the flexibility and control to choose between manual or automated methods to resolve discrepancies
- Improve operations by automatically aligning the operational status of network resources with the same status in the inventory CMDB

Related topics

Discrepancy identification – types of discrepancies

Activate Telecom Discrepancy Identification and Reconciliation

Run Telecom Discrepancy audit

Telecom discrepancy identification and reconciliation

Discrepancy identification – types of discrepancies

The Telecom Discrepancy Identification and Reconciliation capability identifies and classifies mismatches between the network state (as discovered through TSOM Discovery or Service Graph Connectors) and the inventory data stored in the CMDB or TNI.

Discrepancy identification is powered by the CMDB Compliance Certification Audit, which:

- Runs on CI and relationship data.
- Compares discovered and inventoried records.
- Generates Follow-on tasks when mismatches are detected.

Discrepancy types

The audit process identifies discrepancies by comparing discovered network data with the existing inventory in the CMDB/TNI. Discrepancies fall into four categories:

Missing in network - entities existing in inventory but missing in network

Definition: A CI is detected by Discovery but is either missing from the CMDB/TNI or incorrectly represented. For example, Discovery detects Card05 installed in Slot04, but the CMDB still lists Card04, or worse, shows both Card04 and Card05 in the same slot—violating cardinality or model constraints.

Impact:

- A discrepancy task is generated to highlight the data conflict.
- Optional remediation subflows may be triggered to reconcile the data by retiring outdated records or updating slot assignments.

Mismatched configuration items (CIs) - entities that exist in inventory and network but differ in attribute values and hierarchical relationships

Definition: The CI exists in both Discovery and CMDB/TNI, but discrepancies exist in relationships, hierarchy, or attribute values. The following are the sub types:

- Hierarchy Mismatches - Occur when the structural relationships between CIs (e.g., parent-child associations) are inconsistent. The following are the examples:
  - A chassis contains more child cards than allowed by the model definition.
  - A card is incorrectly associated with a slot in the CMDB that does not align with Discovery data. The validation sources are:
    - `cmdb_rel_ci` records for `Contains::Contained by` relationships.
    - `sn_ni_core_network_model_relationship` table for enforcing model-specific constraints.
- Attribute Value Mismatches - Involve discrepancies in CI field-level properties. The common issues are:
  - Outdated or incorrect Discovery Dates.
  - Inaccurate Model Configurations.
  - Invalid Slot Assignments that breach model rules.

The impact is the affected records are marked as failed in audit reports. Follow-On Tasks may invoke context-specific remediation subflows to realign inventory data with the actual network state.

Note: For more information, see Configure attribute value discrepancy in CMDB 360.

Missing in CMDB — entities discovered on the network but not present in inventory

Definition: A CI is discovered on the network through TSOM Discovery or a Service Graph Connector, but no corresponding record exists in the CMDB. This indicates that the equipment is physically present and operational but has not yet been inventoried. For example, Discovery detects a Fortinet FortiGate-80F device on the network, but no matching CI exists in the CMDB.

Impact:

- The audit identifies the CI as new when its creation date matches its discovery date.
- A single follow-on task is created at the equipment level to prompt review. Individual tasks are not generated for each underlying child CI (such as IP addresses) associated with the equipment, keeping the task list focused and actionable.
- The follow-on task includes the short description **New CI has been discovered**, along with the CI reference and associated product model.
- You can enable or disable follow-on task creation for this discrepancy type using a system property.

SD-WAN discrepancy identification

In addition to telecom-specific discrepancies, the Telecom SD-WAN Network Discrepancy Audit identifies mismatches in your SD-WAN inventory. Standard telecom discrepancy detection focuses on CI classes such as slot, card, and interface. The SD-WAN audit targets a different set of CI classes and validates them against rules specific to the SD-WAN data model.

By default, the audit scopes to CIs whose discovery source contains TSOM. The audit compares discovered SD-WAN data with the inventory in the CMDB and generates follow-on tasks when mismatches are detected.

SD-WAN CI classes and validation requirements

CI class	Validation requirements	Relationship type
Group (cmdb_ci_group)	• Company reference • At least 1 Service child • At least 1 Site child	Members::Member of
Site (cmdb_ci_ni_site)	• Company reference • Location reference • Exactly 1 Group parent	Members::Member of
Service (cmdb_ci_network_service_instance)	• Company reference • Exactly 1 Group parent	Members::Member of
Equipment (sn_tsom_core.audit.equipment_tables)	• Company reference • Max 1 Service parent • Max 1 Site parent • At least 1 IP Address child • Each IP Address child has max 1 Equipment parent	• Service/Site: • Contains::Contained by • IP Address: Owns::Owned by

The SD-WAN audit validates the following CI classes. For each CI, the audit checks that required references are populated and that relationships match the expected structure. A CI that meets all requirements is marked as Certified. If one or more requirements are not met, the CI is marked as Failed and a follow-on task is created.

SD-WAN discrepancy types

The SD-WAN audit detects the same categories of discrepancy as the telecom audit, applied to the SD-WAN CI classes listed above:

Missing reference: A required reference (such as Company or Location) is not populated on the CI. A follow-on task is created with the description identifying which reference is missing.

Incorrect relationships: A required relationship is missing or exceeds the allowed count. For example, a Site CI without a Group parent, or an Equipment CI with more than one Service parent. A follow-on task is created with the description identifying the relationship issue.

Missing in CMDB: A CI is discovered on the network but no corresponding record exists in the CMDB. The audit identifies the CI as new when its creation date matches its discovery date, and creates a single follow-on task at the equipment level. For details on this discrepancy type, see the section **Missing in CMDB** above.

Related topics

Activate Telecom Discrepancy Identification and Reconciliation

Run Telecom Discrepancy audit

Telecom Assurance

Telecom assurance monitors network performance, detects faults, and maintains service quality. ServiceNow Telecommunications Service Operations Management software is integrated with existing monitoring tools to consolidate alerts into a single platform, delivering AI-driven insights and automated workflows from fault detection to resolution.

Some key components of ServiceNow AI Platform® Telecommunications Service Operations Management telecom assurance are:

- Alert management: Managing, escalating, and resolving alerts
- Performance management: Analyzing network performance data to optimize service delivery

Related topics

Configure Telecom Assurance

Now Assist for ITOM

Metric Intelligence

External event management via Telecommunications API notifications

Use the Telecommunications API notification to receive the external events that occurring in the customer network system so that you can promptly respond to them in the ServiceNow.

Introduction to Telecommunications API notification

Telecommunications API notification is a feature available in the Telecommunications Alarm Management Open API application. The Telecommunications API notification enables ServiceNow to receive the incoming notifications that occurring in the external network system

and responds to them in a timely manner. It enables the broadcasting of events to the external systems through platform capabilities by eliminating the need for point connections.

Telecommunications API notification receives incoming notifications from the external systems that are subscribed on your network. When the notifications are received from the external system, you can create the events for the responses by using the Event Management application. Based on the collected information, the Event Management provides dashboards showing a consolidated view of all service-impact events.

Telecommunications API notification data model

The Telecommunications API notification enables ServiceNow to receive incoming notifications through the event-driven architectures such as the Publisher/Subscriber (Pub/Sub) subscription model, Hermes, and Kafka Stream Connect. While cloud customers have the flexibility to select between both architectures, on-premise customers are limited to using their own Kafka or Pub/Sub subscription model.

- To learn more about Stream connect for Apache Kafka Stream, see Using Stream Connect for Apache Kafka.
- To learn more about Hermes Messaging Service, see Hermes Messaging Service.

In the Pub/Sub model, incoming notifications are categorized into topics. You use ServiceNow to publish the incoming notifications to these topics, and subscribers (customers) have the flexibility to select the topics to which they want to subscribe. This process enables subscribers to select only those messages that align with their interests. For example, if there are 10 topics for incoming messages from the external system, a customer can opt to subscribe to two of them based on their requirement. Consequently, when notifications are received from the external system, events are generated specifically for the two topics to which the customer has subscribed.

Related topics

Configuring Telecommunications API notifications

System components installed with Telecommunications API notifications

Fault Management: Events and alerts

Fault Management supports the monitoring, detection, and resolution of configuration and performance issues across SD-WAN-managed network devices. It integrates with Event Management to generate alerts and track events automatically.

Fault Management helps you monitor network devices by collecting alerts related to configuration and performance issues. Key features include alerting, customizable event rules, webhook-based notifications, and integration through the Integrations Launchpad.

Alerts are central to detecting and responding to configuration issues. Each alert is generated based on three core attributes:

- Device name: The affected device (for example, IP router, IP switch, access point)
- Alert type: The category of the event
- Alert level: The severity level (informational, warning, critical)

When a monitored event occurs, you can create an alert with these attributes and begin tracking it. You can configure event rules to control what happens next: grouping related alerts, escalating critical ones, or suppressing noise so your team only sees issues that require action. The Integrations Launchpad provides a central place to manage integrations with external systems.

Managing alerts with event rules

Event rules and alert suppression give you control over how alerts are generated and managed. Configure the following options to control alert generation:

- Event rules: Set criteria such as the device name, alert type, severity, and frequency to determine the events and create the appropriate alert.
- Alert suppression: Consolidate multiple alerts from the same device into a single record.
- Alert grouping: Combine related events into a single alert for easier management.
- Correlation IDs: Link related events and alerts with unique identifiers, improving traceability across the event life cycle. These IDs create connections between relevant objects and alerts, enabling you to navigate directly between them.

For instructions on how to configure Event Management solutions, see the following documentation:

- Configure a webhook
- Configure an event pull connector
- Cisco Meraki installed integrations
- Fortinet installed integrations

Performance management: Metric collection

Performance management enables you to capture and analyze operational metrics to identify anomalies. Detected anomalies can generate alerts that surface in Service Operations Workspace, helping you avoid service outages.

Use provided solutions to collect metrics and monitor your devices for different device platforms:

- Cisco Meraki installed integrations
- Fortinet installed integrations

MPN Formula Engine processing

The Formula Engine processes raw KPI formulas and stores the result in the **Formatted KPI Formula** field when a record is inserted or updated in the MPN Formulas [sn_tsom_em_conns_kpi_definitions] table.

When a record is inserted or updated in the MPN Formulas table, a business rule named Transform KPI Formula to Script fires automatically. It invokes the `NokiaMpnFormulaEngineSEP` extension point to process the raw formula before it is stored in the **Formatted KPI Formula** field.

The processing runs in two steps:

1. Cleanup: The raw value in **KPI Formula** is normalized into a JavaScript-evaluable expression:
   - Whitespace and line breaks are collapsed to a single space.
   - `Sum()` wrappers are removed; `Max()` and `Min()` are converted to `Math.max()` and `Math.min()`.
   - `**` exponentiation notation is converted to `Math.pow()` for compatibility with the MID Server JavaScript engine.
2. Validation — The processed formula is checked for balanced parentheses. If the parentheses are unbalanced, the raw value is still stored but a warning is logged. Records with unbalanced parentheses are skipped during metric collection at runtime.

The resulting value is written to the **Formatted KPI Formula** field and is the expression that the metric calculation engine references at runtime.

Note: If no `NokiaMpnFormulaEngineSEP` implementation is registered, the raw value in the **KPI Formula** value is stored unchanged and a warning is logged.

Related topics

MPN Formulas table

Metric-to-CI binding

Metric-to-CI binding associates metrics collected by a pull connector with configuration items (CIs) in the Configuration Management Database (CMDB). Default binding logic is used for each supported connector, which you can override with a custom implementation.

How metric-to-CI binding works

When a metric is collected, the platform runs the following sequence to associate each metric with a CMDB CI:

1. The connector pulls metric data from the vendor and stores it in the metric base (Clotho).
2. If the metric is not yet bound to a CI, the metric base framework automatically creates an event to record the request for a binding.
3. An event field mapping rule fires on the new event. Telecommunications Service Operations Management ships pre-configured event field mapping rules for each supported connector source.
4. The rule delegates to a scripted extension that resolves the appropriate CI for the event. The script updates the `cmdb_ci` field on the event with the resolved CI.
5. The binding appears in the Metric to CI mappings table. From that point on, all subsequent metric data for that source-resource pair is associated with the bound CI directly, without creating another event.

If the scripted extension can't resolve a CI, the event's `cmdb_ci` field is left empty and the metric data is stored in the metric base unbound.

Default behavior for MPN metrics

No additional configuration is required for CI binding. A scripted extension named `NokiaMPNMetricCIMapper` handles metric-to-CI binding for the MPN pull connector by firing on matching events through an event field mapping rule.

The extension tries the following event fields in order, stopping at the first match. The first lookup uses the CMDB table named in `additional_info.ciClass` on the event (defaulting to `cmdb_ci`); the remaining lookups always run against `cmdb_ci`:

MPN metric event CI lookup order

Order	Event field	Lookup behavior
1	`additional_info.name`	Matched against `name` on the class specified by `additional_info.ciClass`
2	`additional_info.pmDataSource.dn`	Distinguished name traversal in `cmdb_ci`. The deepest matching component wins
3	`additional_info.pmDataSource.serialNumber`	Matched against `serial_number` in `cmdb_ci`
4	`additional_info.pmDataSource.name`	Matched against `name` in `cmdb_ci`
5	`additional_info.pmDataSource.name`	Matched against `name` in `cmdb_ci`

The `additional_info.name` and `additional_info.pmDataSource.dn` fields can each hold a comma-separated list of values. The extension tries the values from right to left, and the first value that resolves to a CI wins.

On a match, the extension sets the `cmdb_ci` and `ci_type` fields on the event.

Override the default binding behavior

If the shipped lookup logic does not match how your CMDB models the source data, you can override it by providing your own implementation of the `EventFieldMapping` extension point. The platform resolves the implementation at runtime by source name and rule name, so you can substitute your own logic without modifying product code.

For the procedure, see Override default metric-to-CI binding.

Related topics

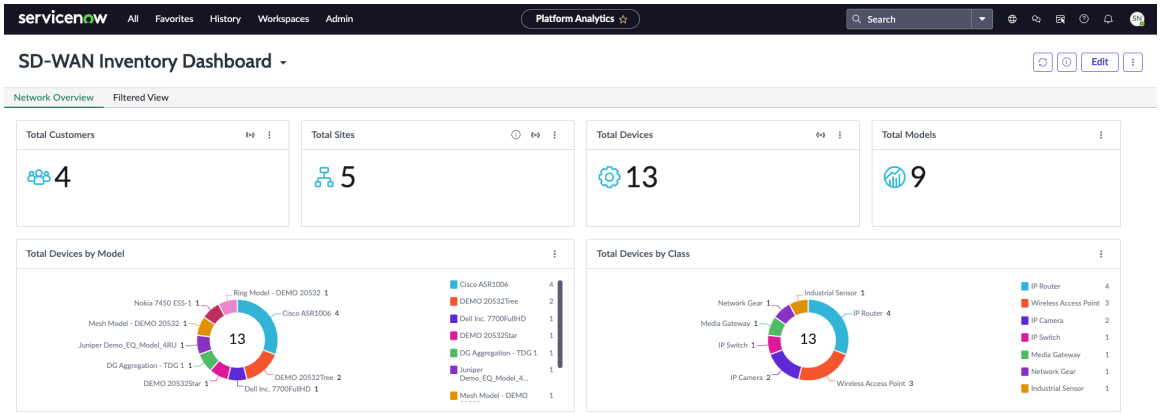
- View metric to CI and resource binding
- Event field mapping configuration
- Create an event rule to bind metric events to host CIs
- Configure elastic event pull connectors for MPN
- View metric values in the Insights Explorer
- Create an Insights Explorer view
- Resource binding

SD-WAN Inventory Dashboard

The SD-WAN Inventory Dashboard provides a visual summary of your SD-WAN network inventory, including total customers, sites, devices, and models, with breakdowns by device model and class.

Accessing the SD-WAN Inventory Dashboard

If you're a user with the tsom_assurance_admin role, you can access the SD-WAN Inventory Dashboard by navigating to **All** > **Platform Analytics > Dashboards** and searching for and selecting SD-WAN Inventory Dashboard.



Network Overview

The **Network Overview** tab provides a high-level summary of all discovered SD-WAN network inventory in the CMDB.

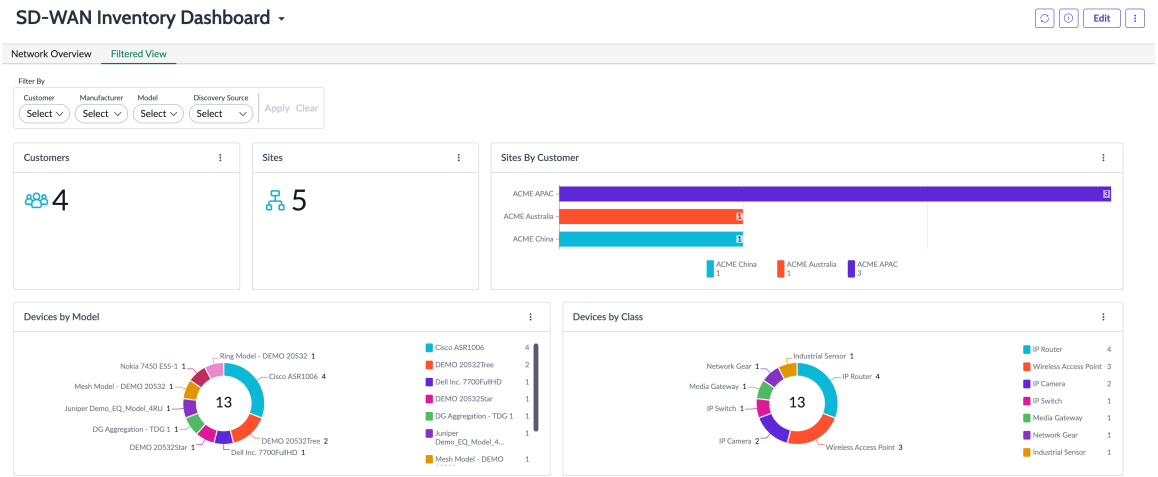
The following breakdowns are available in the SD-WAN Inventory Dashboard:

- Total devices by model
- Total devices by class

Title	Type	Description
Total customers	Single score	The total number of customers with SD-WAN network inventory discovered in the CMDB.
Total Sites	Single score	The total number of sites across all customers with discovered SD-WAN inventory.
Total Devices	Single score	The total number of SD-WAN devices discovered in the CMDB.
Total Models	Single score	The total number of distinct device models discovered across the SD-WAN network.
Total Devices by Model	Donut chart	The distribution of discovered devices by hardware model.
Total Devices by Class	Donut chart	The distribution of discovered devices by device class, such as IP Router, Wireless Access Point, and IP Camera.

Filtered View

The **Filtered View** tab provides the ability to filter inventory data by customer, manufacturer, model, or discovery source.



Title	Type	Description
Customers	Single score	The number of customers matching the applied filter criteria.
Sites	Single score	The number of sites matching the applied filter criteria.
Sites by Customer	Bar Chart	The number of sites per customer matching the applied filter criteria.
Devices by Model	Donut Chart	The distribution of filtered devices by hardware model.
Devices by Class	Donut Chart	The distribution of filtered devices by device class.

SD-WAN Alerts Dashboard

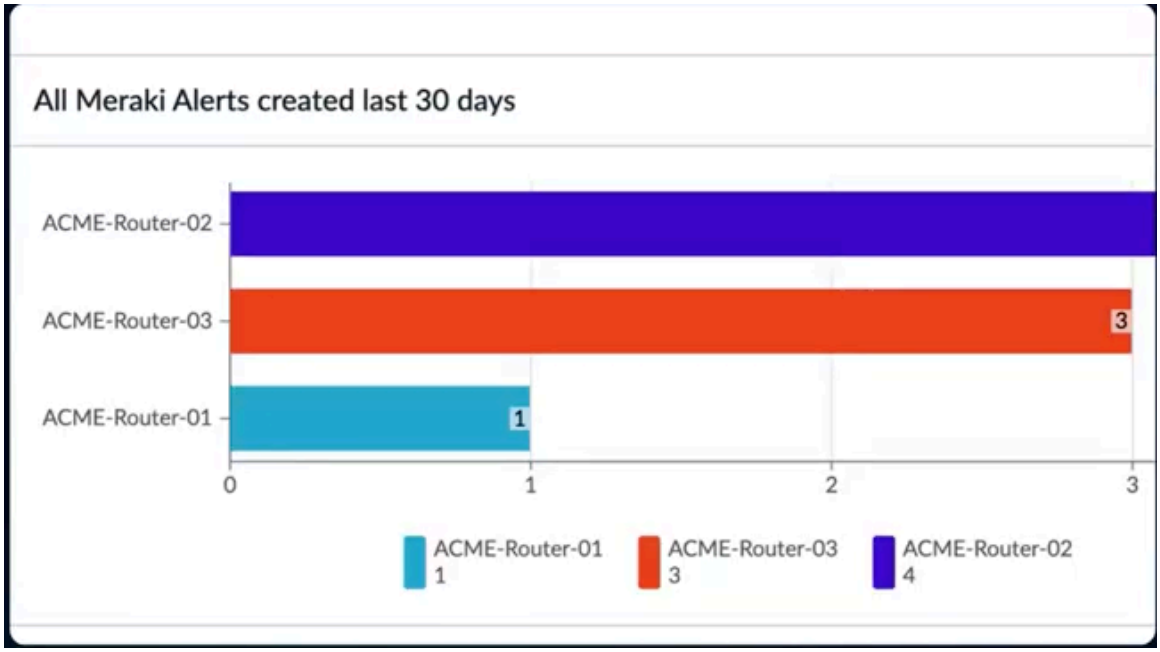
Monitor and analyze SD-WAN network alerts across customers and devices, with views for today's alerts, 30-day alert history, and alert trends over time.

Access the SD-WAN Alerts Dashboard

Access the SD-WAN Alerts Dashboard by navigating to **All** > **Platform Analytics** > **Dashboards** and searching for or selecting SD-WAN Alerts Dashboard. By default, the **Alert Summary** tab is selected.

Alert Summary

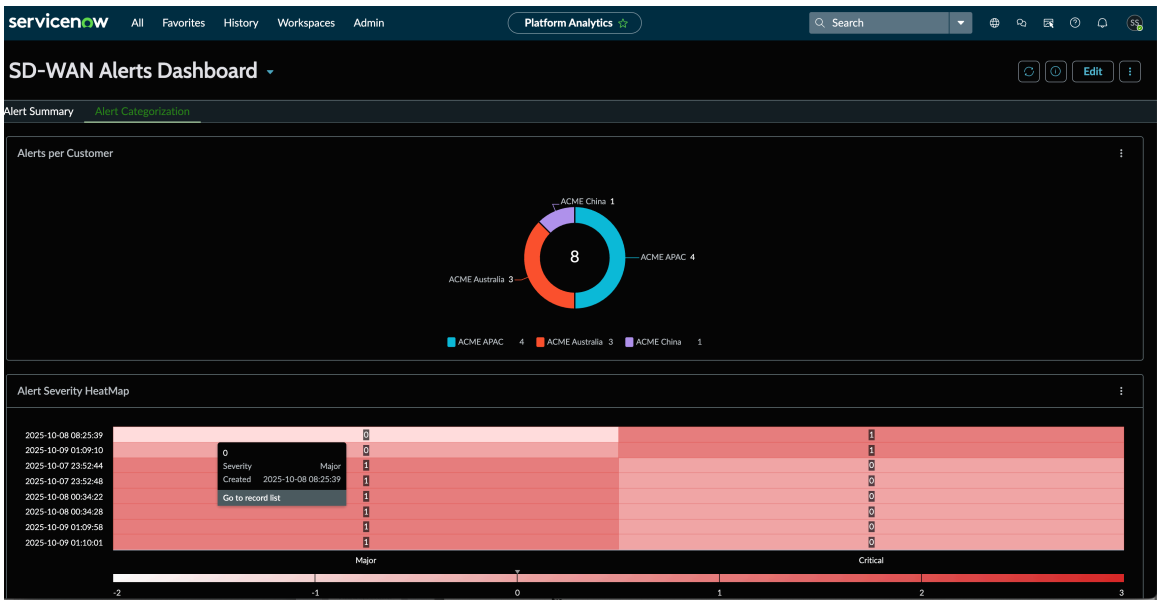
The **Alert Summary** tab provides the capability to filter SD-WAN alert activity by severity, state, customer, or CI.



Title	Type	Description
All alerts created today	Bar chart	The number of alerts created today, grouped by CI
All alerts created the last 30 days	Bar chart	The number of alerts created in the last 30 days, grouped by CI
Alert Trends Over Time	Line graph	The trend of alert activity over time

Alert Categorization

The **Alert Categorization** tab displays SD-WAN alerts grouped by customer and severity.



Title	Type	Description
Alerts per Customer	Donut chart	Distribution of alerts grouped by customer
Alert Severity HeatMap	Heat map	Distribution of alerts by severity level (Major, Minor, and Warning) over time

Configuring Telecommunications Service Operations Management

Configure Telecommunications Service Operations Management (TSOM) to enable real-time event ingestion, correlation, and automated remediation by integrating with external network monitoring and discovery systems.

Set up TSOM to enable end-to-end telecom service operations, including alarm ingestion, CMDB population, discrepancy detection, and service impact visibility. This configuration involves activating Telecommunications API notifications, setting up discovery and service graph connectors, enabling visibility, and configuring audit and reconciliation frameworks.

Configuration overview

Telecommunications Service Operations Management (TSOM) requires configuring multiple components across alarm ingestion, discovery, data normalization, and reconciliation to support end-to-end telecom operations. The configuration flow typically includes the next plugins (The plugin dependency will be automatically resolved during the installation).

1. Install the following plugins:

TSOM plugins

Name	ID	Product/Service
TSOM Event Management Connectors	sn_tsom_em_connectors	Telecom Assurance
Service Graph Connector for Fortinet Telco SD-WAN	sn_tsom_fortinet_connector	Telecom Visibility
Service Graph Connector for Meraki Telco SD-WAN	sn_tsom_meraki_connector	Telecom Visibility
Service Graph Connector for VeloCloud Telco SD-WAN	sn_tsom_vcloud_connector	Telecom Visibility

2. Enable alarm ingestion: Activate the Telecommunications API notifications and set up topic subscriptions to receive alarms from external systems. For more information, see Configuring Telecommunications API notifications.
3. Set up visibility or Event Management by following the appropriate setup wizards. See Configure Telecom Visibility and Fault Management: Events and alerts.

Each step is modular and can be configured based on your environment and available integrations.

Configure Telecom Visibility

Telecom Visibility provides foundational capabilities that support both Telecom Discovery and Telecom Discrepancy Identification & Reconciliation. It includes shared logic, enhanced CI class models, and Identification and Reconciliation Engine (IRE) updates tailored for telecom network elements.

Before you begin

Role required: tsom_visibility_admin

Confirm that:

- Your ServiceNow AI Platform instance is licensed for Telecommunications Service Operations Management.
- You review any custom IRE rules applied to telecom CIs to help prevent overrides during upgrade.

About this task

To configure Telecom Visibility, you must install the required plugins and update the CMDB CI Class Models to version 1.82 or later, which introduces telecom-specific IRE identification rules. This ensures accurate CI identification and reconciliation across telecom domains.

Note: If you have customized IRE identification rules for any of the affected telecom CIs, upgrading to version 1.82 or later may override or impact those rules.

The following capabilities are included with the Telecom Visibility subscription:

Plugin Name	Plugin ID	Plugin Description	Store App
Service Graph Connector for Nokia Altiplano	sn_tsom_altiplano_connector	Service Graph Connector for Nokia Altiplano	Yes
Service Graph Connector for Cisco Meraki Telco SD-WAN	sn_tsom_meraki_connector	Service Graph Connector for Cisco Meraki	Yes
Service Graph Connector for Fortinet	sn_tsom_fortinet_connector	Service Graph Connector for Fortinet	Yes
Service Graph Connector for Arista VeloCloud	sn_tsom_vcloud_connector	Service Graph Connector for Arista VeloCloud	Yes
Telecommunications Discovery Patterns	sn_tsom_patterns	Telecommunication Discovery Patterns	Yes
Telecom Visibility Plugin (Core logic)	sn_tsom_core	Core logic for Telecom Visibility and Discrepancy Reconciliation	No

Procedure

1. Navigate to **All** > **Plugins**.
2. Search for and install the following:
   a. Telecommunication Discovery Patterns (sn_tsom_patterns)
   b. Telecom Visibility (sn_tsom_core)
   c. Service Graph Connector for your vendor.

3. **Optional:** Update the CMDB CI Class Models.
   a. Navigate to **All** > **Applications** list in the ServiceNow Store.
   b. Search for CMDB CI Class Models (sn_cmdb_ci_class).
   c. Install or upgrade to version 1.82 or later.

   Note: Installing any of the TSOM plugins listed above will automatically update or install version 1.82 or later of the CMDB CI Class Models app. If your instance doesn't include TSOM plugins or you're on a pre-Yokohama release (for example, Washington DC or Xanadu), you can manually install or upgrade the store app.

4. Navigate to **CMDB** > **Identification Rules**.
   a. Review rules related to:
      - `cmdb_ci_interface_card`
      - `cmdb_ci_slot`
      - `cmdb_ci_subslot`
      - `cmdb_ci_network_adapter`
   b. Validate that your custom logic remains functional.
   c. Update or merge custom rules as needed.

Result
Confirm that the following are listed as active:

- Telecom Visibility (sn_tsom_core)
- Telecommunication Discovery Patterns (sn_tsom_patterns)
- Service Graph Connector for your vendor
- CMDB CI Class Models version 1.82 or later

Related topics
Telecom Visibility

Install Horizontal Discovery and set up Discovery Patterns

Install Horizontal Discovery patterns understanding the dependencies and requirements.

Before you begin
Role required: admin

Ensure you have a subscription to TSOM.

About this task
Visit the ServiceNow Store website to view all the available apps and for information about submitting requests to the store. For cumulative release notes information for all released apps, see the ServiceNow Store version history release notes.

Dependencies and Requirements:

- Telecom Visibility Core (sn_tsom_core)
- Discovery Core plugin (com.snc.discovery.core), which is automatically installed by Discovery.

- ITOM Discovery License plugin (com.snc.itom.discovery.license). You must activate this plugin.
- ITOM Licensing plugin (com.snc.itom.license). For more information, see Request Discovery.

Pattern execution logic: By default, when a TSOM pattern is run, it executes both the TSOM-specific and the corresponding ITOM pattern (e.g., Telco Router runs Router). This ensures shared ITOM libraries are reused when needed. To override this behavior:

- Use the system property sn_tsom_patterns.itom_pattern_enabled.
- Setting this property to false ensures only the TSOM-specific pattern is executed.

Procedure

1. Install the Horizontal Discovery application.
   See Discovery setup, as it is foundational for running Telecommunications Discovery Patterns.
2. Obtain and install Telecommunications Discovery Patterns:
   a. Install the Telecommunications Discovery Patterns (sn_tsom_patterns) from the ServiceNow® Store.
3. Set up a MID Server and synchronization Patterns:
   a. Synchronization the installed patterns with the appropriate MID Servers to confirm they're ready for use:
      i. Navigate to **Discovery** > **MID Servers**.
      ii. Select **Pattern Sync to Mid**.

      Note: This action synchronizes both TSOM and ITOM patterns.

      For more information on how to configure a MID Server, see Configuring MID Server.
4. Configure TSOM System Properties:
   a. Set the system property sn_tsom_patterns.itom_pattern_enabled to define the logic for whether to use only the TSOM Pattern or a combination of ITOM and TSOM patterns.
      i. Navigate to**All** > **System Properties** > **All Properties**.
      ii. Select **sn_tsom_patterns.itom_pattern_enabled**.
      iii. Check that the Value is set to **true** (default).

      If you want TSOM to run only TSOM patterns and exclude ITOM patterns, set the Value to **false**.

      Note: The default setting is configured to use both TSOM and ITOM patterns.
5. Enable the replacement of various ITOM patterns with TSOM patters on a specific MID Server:

   For example: The Telecom Router pattern replaces the Network Router pattern for a specific MID Server when **mid.telecom.discovery.patterns.enabled** is set to true for that MID Server.

a. Go to the **Filter Navigator** and type **ecc_agent_config.list**.

b. Select **mid.telecom.discovery.patterns.enabled** (each MID Server has this parameter).

c. Check that the Value is set to **true**.

Repeat this configuration for each MID Server that you want to use for running TSOM patterns.

Related topics

Direct Discovery using Discovery Patterns

Activate Telecom Discrepancy Identification and Reconciliation

Activate the Telecom Discrepancy Identification and Reconciliation feature (part of the Telecom Visibility plugin) to ensure consistency between telecom network resources discovered in the live network and the data represented in the CMDB or Telecom Network Inventory (TNI). This feature helps detect and remediate mismatches automatically, supporting service accuracy and data integrity across your telecom environment.

Before you begin

Role required: admin

To use this feature, your organization must have an active subscription to TSOM. Telecom Discovery, Telecom Visibility, and this feature are licensed together.

About this task

Plugin dependencies - Ensure the following plugins are installed and activated:

Plugin	ID / App ID	Type
Telecom Service Operation Core	`sn_tsom_core`	Store
CMDB CI Class Models	`sn_cmdb_ci_class`	Store
Expanded Model and Asset Classes	`sn_ent`	Store
Visibility Content	`sn_pattern_design`	Store
Integration Commons for CMDB	`sn_cmdb_int_util`	Store
Discovery Core	`com.snc.discovery.core`	Family
ITOM Discovery License	`com.snc.itom.discovery.license`	Family
ITOM Licensing	`com.snc.itom.license`	Family

Procedure

1. Ensure the sn_tsom_core plugin is automatically installed when you install Telecommunications Discovery Patterns or the Nokia Altiplano Service Graph Connector.
2. Ensure the mentioned dependent plugins are activated.

Related topics

Discrepancy identification – types of discrepancies

Telecom discrepancy identification and reconciliation

Telecom Discrepancy Identification and Reconciliation

Run Telecom Discrepancy audit

Control CI attribute updates using Reconciliation rules

Configure filter for audit

Define filtering conditions to control the scope of Telecom Discrepancy audits. These filters ensure that audits run only on the desired set of CIs, improving performance and targeting accuracy.

Before you begin

Role required: admin

About this task

Understanding Filtering Conditions in Telecom Discrepancy Audits

- Audits use filtering conditions to narrow the scope of CIs being evaluated.
- Filtering conditions are essential for narrowing down audit targets.
- You can define and modify conditions per audit instance to meet specific audit requirements.
- Filtering by `discovery_source` is a common condition to filter records associated with particular integration sources (e.g., TSOM and Altiplano). Additionally, you can customize the filter according to the audit requirements.
- Each version of the audit can have a different filtering condition.

The following are the audit filters:

1. Telecom Logical Connections Discrepancy Audit: This audit runs at the logical connection table. The default filter condition is defined as `discovery_source` like TSOM. This condition is applied on each CI in the logical connection table.
2. Telecom Network Topology Discrepancy Audit: This audit runs at the network topology table. The default filter condition is defined as `discovery_source` like TSOM. This condition is applied on each CI in the network topology table.
3. Telecom Discrepancy Audit: This audit runs at the relationships table. The default filter condition is defined as `discovery_source` like TSOM. This condition is applied on each CI at the equipment level.

Procedure

1. Navigate to **All** > **Certification** > **Filters**.
2. Select **New**.
   The certification filter interface appears.
3. Fill in the fields.
   For more guidance, see Create a filter.
4. Add the filter conditions.
   For example, Discovery source contains TSOM.
5. Select **Submit**.
   The filter is created.

Result

You can use the filter in the logical connection audit.

Related topics

- Run Telecom Discrepancy audit
- Example for Telecom Discrepancy Audit and Remediation
- Telecom discrepancy identification and reconciliation
- Configure attribute value discrepancy in CMDB 360
- Control CI attribute updates using Reconciliation rules

Configure attribute value discrepancy in CMDB 360

Configure attribute comparison settings in CMDB 360 to detect data inconsistencies across multiple discovery sources.

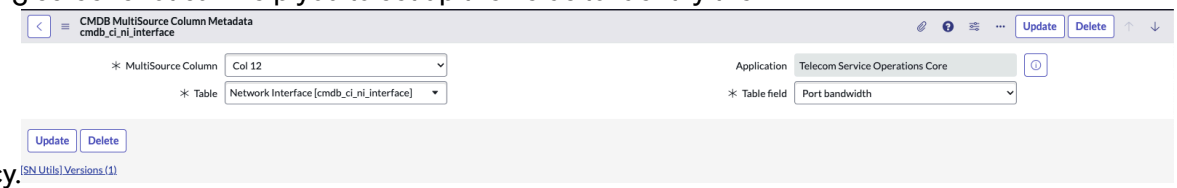
Before you begin

Role required: admin

Confirm that CMDB 360 is enabled and configured in your instance.

About this task

The following screenshot can help you to set up the fields to identify the discrepancy.



Procedure

1. Navigate to the table **All** > **cmdb_multisource_column_metadata.list**.
2. Select **New** to create a new record.
3. In the **MultiSource Column** field, select the attribute to compare.
4. Select the table to compare the attributes to the fields.

> **Note:** If the table you need isn't listed, select and hold (or right-click) on the Table label and select **Configure Dictionary**.

5. **Optional:** To configure a dictionary entry, do the following.
   a. In the dictionary entry, select **View** and then select **Advanced**.
   b. In the **Attributes** field, set the attributes as `base_start=true` and `allow_public=true`.

Result

You can use the configured attributes in the CMDB 360 query.

Related topics

- Control CI attribute updates using Reconciliation rules
- Generate reports for attribute value discrepancies

Control CI attribute updates using Reconciliation rules

To prevent specific attributes of a Configuration Item (CI) from being overwritten by Discovery or other data sources, use Reconciliation Rules. These rules define which data source is trusted to update a particular attribute when multiple sources provide values.

Before you begin
Role required: admin

About this task
Reconciliation Rules are processed by the Identification and Reconciliation Engine (IRE) and are essential for maintaining data integrity in the CMDB.

Use Case: If you want to prevent Discovery from updating a set of attributes while allowing another source (like SCCM or manual entry) to update them, define a rule that excludes the Discovery source.

Note: Only one Reconciliation Rule should be active for a specific CI class and attribute combination to avoid conflicts.

Procedure
1. Navigate to **All** > **CI Class Manager**.
2. Select the target CI class (e.g., cmdb_ci_computer).
3. In the **Reconciliation Rules** tab, click **New** to create a rule.
4. Provide the rule name and select the Discovery source to apply the rule to.
5. Specify the attributes that this rule will govern.
6. Set the source precedence to allow only the selected sources to update the specified attributes.

Result
Certain CI attributes will no longer be updated by untrusted or lower-priority data sources like Discovery. Only the sources defined in the rule (e.g., SCCM, manual updates) will be allowed to update those attributes.

Related topics
Configure attribute value discrepancy in CMDB 360

Configure the Telecom Discovery Builder ETL

Leverage the prebuilt Telecom Discovery Builder framework ETL by duplicating it into your connector scope, assigning the appropriate data source, and deploying it as part of a new service graph connector.

The Telecom Discovery Builder framework ETL provided with the Telecom Service Operations Management (TSOM) Core is a ready-to-use framework designed to streamline data ingestion into the CMDB across telecom connectors. Rather than configuring it directly, as an administrator, you can duplicate the ETL into a connector's application scope and update the data source to align with the connector's discovery payload.

Steps to use the Telecom Discovery Builder framework ETL in a connector

1. Create a temporary data source: Create a data source in your connector's application scope based on the Generic Schema multi source data source from TSOM Core. This enables you to run and test the duplicated ETL with schema-aligned placeholder data. For more information, see Create a data source similar to Telecom core data source
2. Duplicate the Telecom Discovery Builder framework ETL: Access ETL Studio, locate the Telco Generic Schema ETL, and duplicate it into the connector's scope. During duplication: Provide a new name. Assign the temporary data source. Use importSet as the discovery source. For more information, see Duplicate the Telecom Discovery Builder ETL

3. After duplication, open the new ETL and replace the temporary data source with the connector's actual discovery data source. For more information, see Update the data source of the connector.
4. Test or deploy the service graph connector. For more information, see Deploy service graph connector with existing ETL

Create a data source similar to Telecom core data source

Set up a schema-compliant data source in the connector's application scope to simulate telecom data and ensure successful testing and validation of the Telecom Discovery Builder framework ETL before integrating with live device data.

Before you begin

Role required: admin

Ensure the following:

- Access to the TSOM Core application and its data sources.
- Switch to the connector's application scope.
- Have a sample payload that conforms to the Telco Generic Schema (optional, but recommended for testing).

About this task

When duplicating the Telecom Discovery Builder framework ETL into a Service Graph Connector (SGC) application scope, you must first create a data source that replicates the exact structure of the TSOM Core data source. This duplicated data source provides the required schema and structure for testing and validating the ETL before connecting it to live telecom data.

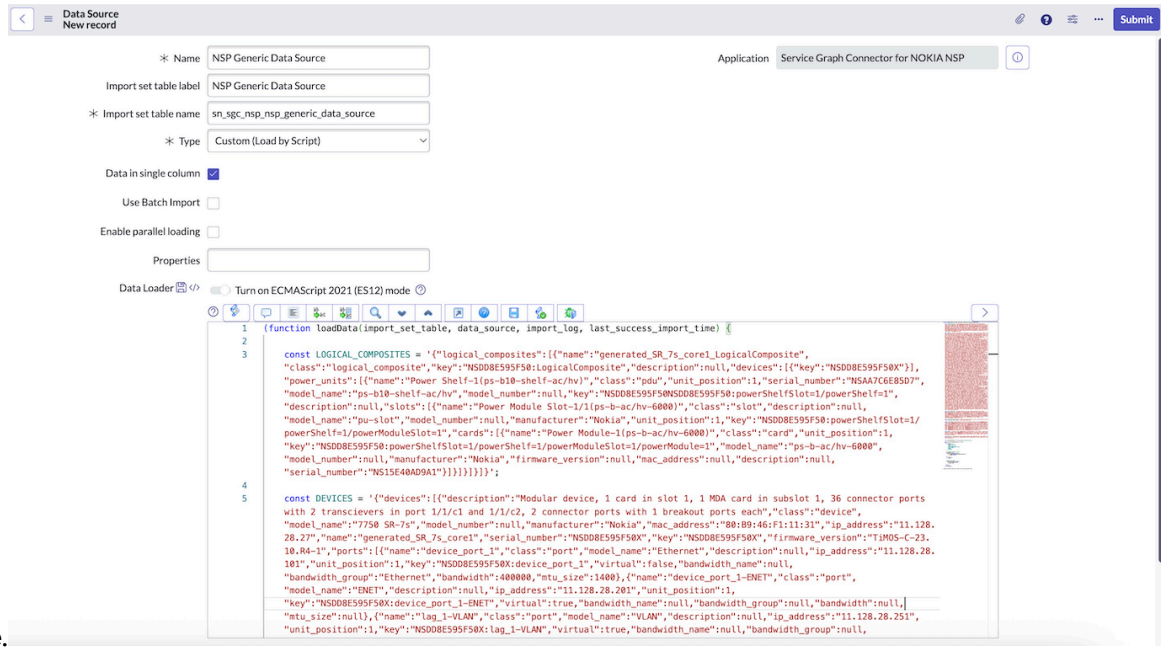
The Telecom Discovery Builder framework expects data that conforms to the Telco Generic Schema, as defined in the Telecom Core. Duplicating the original TSOM data source ensures:

- Schema alignment during ETL duplication.
- A valid import set structure for testing.
- Separation of core and connector scopes for customization and upgrade safety.

When to create a data source similar to the TSOM Core data source:

- Before duplicating the Telecom Discovery Builder framework ETL.
- When you want to run a test load using simulated or placeholder telecom data.
- When preparing the connector's application scope for ingestion configuration.

The following screenshot can help you understand to fill the field values while you create the data source.



Procedure

1. Navigate to **All** > **System Import Sets** > **Administration** > **Data Sources**.
2. Find the **Generic Schema Multi Source v2** or a similar baseline source provided with TSOM Core.
3. Open the TSOM Core data source record and copy the script from the **Data Loader** field.
4. Create a data source by clicking **New**.
5. On the form, fill in the fields
   For more information, see Create a Custom (Load by Script) type data source.
6. In the **Type** field, select **Custom (Load by Script)**.
7. Select the **Data in single column** field.
8. In the **Data Loader** field, paste the copied script.
9. Select **Submit**.
   The data source is created.
10. **Optional:** To test load the data source:
    a. Click Test Load 20 Records (or similar) to generate an import set.
    b. Ensure that records are created without errors.
    c. Confirm that classes like Logical Composite, Network Gear, or Port appear in the staging table.

What to do next

After the data source is created and tested:

- Use it as the import source when duplicating the Telco Generic ETL.
- Replace it with the actual connector-specific data source once simulation and validation are complete.

Related topics

Standardized JSON data set for service graph connectors

Duplicate the Telecom Discovery Builder ETL

Duplicate the Telecom Discovery Builder ETL

The Telecom Discovery Builder framework ETL enables Service Graph Connector (SGC) teams to rapidly adopt a standardized, schema-compliant data ingestion pipeline without building ETL logic from scratch.

Before you begin

Role required: admin

Ensure that:

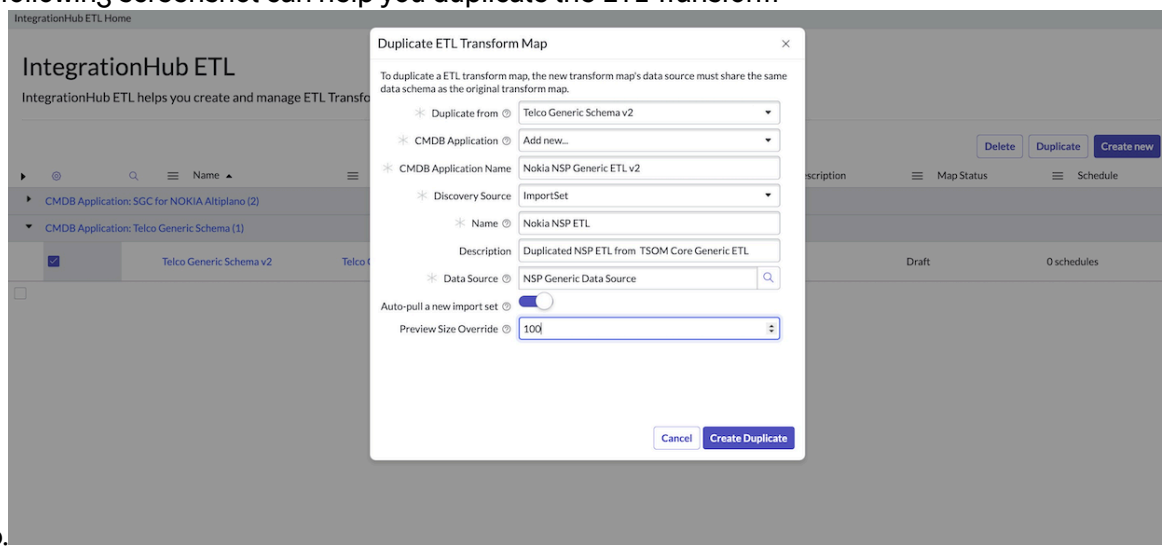
- The Telecom Core plugin is activated and the Telco Generic ETL v2 (automatically installed) is available.
- You have admin access to IntegrationHub ETL Studio.
- You have created a temporary data source in the connector's application scope. After creating a temporary data source, you can duplicate the Generic ETL into your connector's application scope to customize and extend it for your specific use case.

About this task

Duplicating the Telco Generic ETL allows you to:

- Reuse standardized ETL mappings across multiple connectors.
- Customize ETL behavior without altering the original baseline provided by the Telecom Service Operations Management (TSOM) Core.
- Align with the Telco Generic Schema for consistency and TNI compliance.
- Save time and reduce errors by working with a tested and proven ETL framework.

The following screenshot can help you duplicate the ETL Transform Map.



Procedure

1. Navigate to **All** > **IntegrationHub ETL**.
2. On the **IntegrationHub ETL** page, select the connector's application scope.

> **Note:** For example, **Service Graph Connector for Nokia NSP**.

3. Expand the CMDB Application Telco Generic Schema and select **Telco Generic Schema** record.
4. Click **Duplicate**.
   The Duplicate ETL Transform Map page appears.
5. Configure the duplicated ETL transform map

   a. In the **Duplicate From** field, ensure Telco Generic Schema is selected.

   b. Select an existing CMDB Application or click **Add new...** to add a new CMDB Application.

   c. For a new CMDB Application, enter the CMDB application name (for example, Nokia NSP Generic ETL v2)

   d. In the **Discovery Source** field, select the import set.

   e. In the **Name** field, enter a name for the duplicate ETL.

   f. In the **Description** field, enter a description for the duplicate ETL.

   g. Select the data source that is used for the duplicate ETL transform map.

   > **Note:** This needs to be different than the existing default Data Source that is attached to the Telco Generic Schema ETL. For more information, see Create a data source similar to Telecom core data source.

   h. Enable **Auto-pull a new import set** option to automatically pull data into a new import set.

   i. In the **Preview Size Override** field, set a custom preview size for testing and validation.

6. Select **Create Duplicate**.
   A duplicate ETL of Telco Generic ETL is created.
7. **Optional:** Open the duplicated ETL record and review the mappings and settings.

What to do next

1. Test the duplicated ETL (optional but recommended):
   - Run a test load or a simulation using the temporary data source.
   - Verify that import sets are processed successfully, CIs are created according to the Telco Generic Schema, and relationships are established properly.
2. After successful duplication and testing:
   - Update the ETL's data source configuration to point to the actual production data source (for real device data).
   - Deploy the connector integration into a test or production environment.
   - Monitor import runs to validate that inventory data is ingested into the CMDB correctly.

Related topics

Update the data source of the connector

Example - Duplicate the Telco Generic ETL Schema

This example walks you through how to duplicate the Telco Generic Schema ETL to set up a customized Service Graph Connector (SGC) ETL for your telecom integration. Use this procedure when you want to create a baseline ETL in your connector's application scope based on the standardized Telco Generic Schema, ensuring schema alignment, consistency, and faster deployment.

Scenario

You are deploying a new Service Graph Connector and need to duplicate the Telco Generic Schema ETL into your connector's application scope.

Create a temporary data source, duplicate the ETL, and configure it to work with your connector's device payloads.

Steps to duplicate the Telco Generic Schema ETL

1. Switch to the connector's application scope where you want to duplicate the ETL (e.g., Nokia NSP Connector).
2. Create the data source:
   - Navigate to System Import Sets > Administration > Data Sources.
   - Locate the Generic Schema Multi-Source Data Source provided by the TSOM Core application.
   - Copy this data source into your connector's application scope.
   - Test the copied data source by running Test Load 20 Records to create a sample import set.
3. Open the Duplicate ETL Transform Map Dialog: In ETL Studio, select Duplicate ETL to start the duplication process.
4. In the **Duplicate from** list, choose Telco Generic Schema.
5. Select **Add new...** and enter a name for the duplicated ETL.
6. Select importSet as the discovery source for your duplicated ETL.
7. Enter a new name for the duplicated transform map.
8. Specify the newly created temporary data source that you copied from the Generic Schema Multi Source.
9. Optionally, enable **Auto-pull a new import set** to automatically load new records after duplication.
10. Click **Create Duplicate** to complete the duplication.
11. Update **Basic Details**:
    - After duplication, open the newly created ETL.
    - In step 1: specify **Basic Details**, replace the temporary data source with your connector's production data source.
12. Save the ETL Configuration: click **Save** to finalize your changes.
13. Run the **Data Source**: From Import Schedules, run your connector's data source to ingest real device data.

Result: The system creates a new ETL based on the Telco Generic Schema settings and processes the payloads from your specified connector data source. The imported data is transformed into Configuration Items (CIs) and inserted into the ServiceNow CMDB with the expected relationships and structures.

Update the data source of the connector

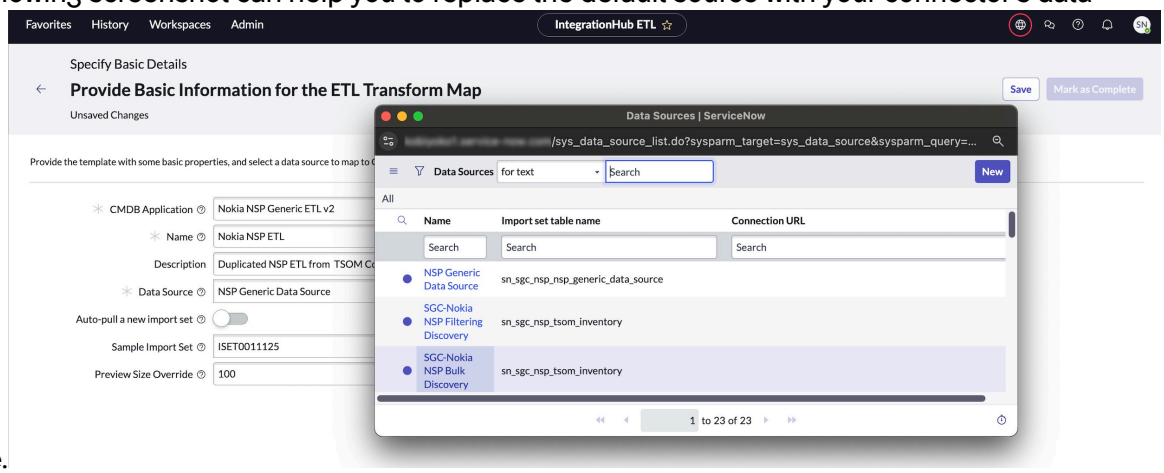
Link the duplicated ETL to a valid data source for your specific connector.

Before you begin

Role required: admin

About this task

The following screenshot can help you to replace the default source with your connector's data source.



Procedure

1. Navigate to **All** > **IntegrationHub ETL**.
2. In **Specify Basic Details**, replace the default data source with your connector's data source.
3. Select **Save**.
4. Open **Import Schedules** and run your data source.
   The system creates a new transform map based on the selected settings and create CIs generated from your payload.

Related topics

Deploy service graph connector with existing ETL

Deploy service graph connector with existing ETL

Associate a new Service Graph Connector (SGC) with an existing ETL.

Before you begin

Role required: admin

Ensure the required Service Graph Connector plugin is activated.

Procedure

1. Navigate to **All** > **IntegrationHub ETL**.
2. Open the duplicated ETL configuration.
3. Select the data source of the service graph connector and click **Save**.
4. Select **Import Schedules** to execute the ETL to transform and load the data from the new SGC.
5. Monitor execution status.

Result

Verify that CI records and relationships are created in the CMDB using CMDB Maps or list views.

Extend TNI entity support for duplicated ETLs

When Telecom Network Inventory (TNI) is enabled, every Configuration Item (CI) created by a duplicated ETL must have a corresponding TNI Entity.

Before you begin

Role required: admin

- Ensure TNI is installed and active in your instance.
- Complete the duplication of the Telecom Discovery Builder framework ETL into the target application scope.

Procedure

1. Navigate to **All** > **IntegrationHub ETL**.
2. Select the CMDB Application associated with the duplicated ETL.
   The CMDB Integration Studio application in a new page
3. In the ETL configuration page, check the **Execute Before Script** option.
4. Replace the default script with the following:

```
(function(input, runId) {
    new
 sn_tsom_core.TelcoGenericMappingHelper().checkAndUpdateIrePayl
oadForTni(input);
})(input, runId);
```

5. Click **Update** to save the changes.
   The duplicated ETL links TNI entities for discovered CIs.

Standardized JSON data set for service graph connectors

Use the TSOM architecture to support standardized Service Graph Connectors using a unified schema and reusable ETL logic. This reduces onboarding time for new connectors and simplifies integration with CMDB.

1. Define the common JSON Schema

Standardize the output of all service graph connectors to align with a single JSON format that conforms to the TNI schema.

Ensure the following points:

1. Implement conversion logic in the collector or adaptor to output data in the common schema.
2. Ensure:
   - Slot-on-slot or card-on-card hierarchies are excluded.
   - Logical interfaces are clearly marked with `virtual=true`.
   - Equipment types align to model-class mappings.

Note: The schema should support runtime adaptability to TNI changes if available.

The following is the JSON schema common data set that supports all TSOM service graph connectors:

```
{
  "$schema": "http://json-schema.org/draft-07/schema#",
```

```
"title": "Telco Generic Schema",
"version": "2.0.1",
"oneOf": [
  {
    "type": "object",
    "properties": {
      "logical_composites": {
        "type": "array",
        "items": {
          "$ref": "#/$defs/logical_composite"
        }
      }
    },
    "required": [
      "logical_composites"
    ],
    "additionalProperties": false
  },
  {
    "type": "object",
    "properties": {
      "devices": {
        "type": "array",
        "items": {
          "$ref": "#/$defs/device"
        }
      }
    },
    "required": [
      "devices"
    ],
    "additionalProperties": false
  },
  {
    "type": "object",
    "properties": {
      "logical_connections": {
        "type": "array",
        "items": {
          "$ref": "#/$defs/logical_connection"
        }
      }
    },
    "required": [
      "logical_connections"
    ],
    "additionalProperties": false
  },
  {
    "type": "object",
    "properties": {
      "port_relations": {
        "type": "array",
        "items": {
          "$ref": "#/$defs/port_relation"
        }
      }
```

```
      },
      "required": [
        "port_relations"
      ],
      "additionalProperties": false
    },
    {
      "type": "object",
      "properties": {
        "logical_connection_relations": {
          "type": "array",
          "items": {
            "$ref": "#/$defs/logical_connection_relation"
          }
        }
      },
      "required": [
        "logical_connection_relations"
      ],
      "additionalProperties": false
    },
    {
      "type": "object",
      "properties": {
        "numbers": {
          "type": "array",
          "items": {
            "$ref": "#/$defs/number"
          }
        }
      },
      "required": [
        "numbers"
      ],
      "additionalProperties": false
    },
    {
      "type": "object",
      "properties": {
        "topologies": {
          "type": "array",
          "items": {
            "$ref": "#/$defs/network_topology"
          }
        }
      },
      "required": [
        "topologies"
      ],
      "additionalProperties": false
    },
    {
      "type": "object",
      "properties": {
        "topology_relations": {
          "type": "array",
          "items": {
```

```
            "$ref": "#/$defs/network_topology_relation"
          }
        }
      },
      "required": [
        "topology_relations"
      ],
      "additionalProperties": false
    }
  ],
  "$defs": {
    "keyRef": {
      "type": "object",
      "properties": { "key": { "type": "string" }
      },
      "required": [ "key" ],
      "additionalProperties": false
    },
    "optionalKeyRef": { "type": [ "object", "null" ],
      "properties": {
        "key": { "type": "string" }
      },
      "additionalProperties": false
    },
    "value": {
      "type": "object",
      "properties": {
        "from": { "type": [ "integer" ], "default": 0,
"minimum": 0 },
        "to": { "type": [ "integer" ], "default": 0, "minimum":
0 }
      },
      "required": [ "from", "to" ],
      "additionalProperties": false
    },
    "logical_composite": {
      "type": "object",
      "properties": {
        "class": { "type": "string", "enum":
[ "logical_composite" ] },
        "key": { "type": "string" },
        "name": { "type": [ "string", "null" ] },
        "description": { "type": [ "string", "null" ] },
        "devices": { "type": "array", "items": { "$ref":
"#/$defs/keyRef" } },
        "power_units": { "type": "array", "items": { "$ref":
"#/$defs/pdu" } },
        "fan_shelves": { "type": "array", "items": { "$ref":
"#/$defs/fan_shelf" } } },
      "required": [ "key", "name" ]
    },
    "pdu": {
      "type": "object",
      "properties": {
        "class": { "type": "string", "enum": [ "pdu" ] },
        "key": { "type": "string" },
        "name": { "type": [ "string", "null" ] },
```

```
        "description": { "type": [ "string", "null" ] },
        "model_name": { "type": [ "string", "null" ] },
        "model_number": { "type": [ "string", "null" ] },
        "unit_position": { "type": [ "integer", "null" ],
"minimum": 1 },
        "slots": { "type": "array", "items": { "$ref":
"#/$defs/slot" } }
      },
      "required": [ "key", "name" ]
    },
    "fan_shelf": {
      "type": "object",
      "properties": {
        "class": { "type": "string", "enum": [ "fan_shelf" ] },
        "key": { "type": "string" },
        "name": { "type": [ "string", "null" ] },
        "description": { "type": [ "string", "null" ] },
        "slots": { "type": "array", "items": { "$ref":
"#/$defs/slot" } }
      },
      "required": [ "key", "name" ]
    },
    "device": {
      "type": "object",
      "properties": {
        "class": { "type": "string", "enum": [ "device" ] },
        "key": { "type": "string" },
        "name": { "type": [ "string", "null" ] },
        "description": { "type": [ "string", "null" ] },
        "ip_address": { "type": [ "string", "null" ] },
        "mac_address": { "type": [ "string", "null" ] },
        "serial_number": { "type": [ "string", "null" ] },
        "model_name": { "type": [ "string", "null" ] },
        "model_number": { "type": [ "string", "null" ] },
        "manufacturer": { "type": [ "string", "null" ] },
        "firmware_version": { "type": [ "string", "null" ] },
        "slots": { "type": "array", "items": { "$ref":
"#/$defs/slot" } },
        "ports": { "type": "array", "items": { "$ref":
"#/$defs/port" } } },
      "required": [ "key", "name", "serial_number" ]
    },
    "slot": {
      "type": "object",
      "properties": {
        "class": { "type": "string", "enum": [ "slot" ] },
        "key": { "type": "string" },
        "name": { "type": [ "string", "null" ] },
        "description": { "type": [ "string", "null" ] },
        "model_name": { "type": [ "string", "null" ] },
        "model_number": { "type": [ "string", "null" ] },
        "manufacturer": { "type": [ "string", "null" ] },
        "unit_position": { "type": [ "integer", "null" ],
"minimum": 1 },
        "cards": { "type": "array", "items": { "$ref":
"#/$defs/card" } }
      },
```

```
      "required": [ "key", "name" ]
    },
    "card": {
      "type": "object",
      "properties": {
        "class": { "type": "string", "enum": [ "card" ] },
        "key": { "type": "string" },
        "name": { "type": [ "string", "null" ] },
        "description": { "type": [ "string", "null" ] },
        "mac_address": { "type": [ "string", "null" ] },
        "serial_number": { "type": [ "string", "null" ] },
        "firmware_version": { "type": [ "string", "null" ] },
        "model_name": { "type": [ "string", "null" ] },
        "model_number": { "type": [ "string", "null" ] },
        "manufacturer": { "type": [ "string", "null" ] },
        "unit_position": { "type": [ "integer", "null" ],
"minimum": 1 },
        "slots": { "type": "array", "items": { "$ref":
"#/$defs/slot" } },
        "ports": { "type": "array", "items": { "$ref":
"#/$defs/port" } }
      },
      "required": [ "key", "name" ]
    },
    "port": {
      "type": "object",
      "properties": {
        "class": { "type": "string", "enum": [ "port" ] },
        "key": { "type": "string" },
        "name": { "type": [ "string", "null" ] },
        "description": { "type": [ "string", "null" ] },
        "model_name": { "type": [ "string", "null" ] },
        "ip_address": { "type": [ "string", "null" ] },
        "virtual": { "type": "boolean", "default": false },
        "unit_position": { "type": [ "integer", "null" ],
"minimum": 1 },
        "bandwidth_name": { "type": [ "string", "null" ] },
        "bandwidth_group": { "type": [ "string", "null" ] },
        "bandwidth": { "type": [ "integer", "null" ], "minimum":
0 },
        "mtu_size": { "type": [ "integer", "null" ], "minimum":
0 }
      },
      "required": [ "key", "name" ]
    },
    "number": {
      "type": "object",
      "properties": {
        "class": { "type": "string", "enum": [ "number" ] },
        "key": { "type": "string" },
        "name": { "type": [ "string", "null" ] },
        "related_ci_type": { "type": "string", "enum":
[ "Network Interface", "Physical Connection", "Logical
Connection", "Equipment", "Topology" ] },
        "related_ci": { "$ref": "#/$defs/keyRef" },
        "type": { "type": "string", "enum": [ "vlan_range",
"vlan_subrange", "vlan", "lag_range", "lag" ] },
```

```
        "vlan_type": { "type": "string", "enum": [ "inner",
"outer" ] },
        "value": { "$ref": "#/$defs/value" }
      },
      "required": [ "key", "name", "type", "related_ci_type",
"related_ci", "value" ] },
    "logical_connection": {
      "type": "object",
      "properties": {
        "class": { "type": "string", "enum":
[ "logical_connection" ] },
        "key": { "type": "string" },
        "name": { "type": [ "string", "null" ] },
        "description": { "type": [ "string", "null" ] },
        "model_name": { "type": [ "string", "null" ] },
        "bandwidth_group": { "type": [ "string", "null" ] },
        "bandwidth_name_a_to_z": { "type": [ "string",
"null" ] },
        "bandwidth_name_z_to_a": { "type": [ "string",
"null" ] },
        "bandwidth_a_to_z": { "type": [ "integer", "null" ],
"minimum": 0 },
        "bandwidth_z_to_a": { "type": [ "integer", "null" ],
"minimum": 0 },
        "equipment_a": { "$ref": "#/$defs/optionalKeyRef" },
        "equipment_z": { "$ref": "#/$defs/optionalKeyRef" },
        "port_a": { "$ref": "#/$defs/keyRef" },
        "port_z": { "$ref": "#/$defs/keyRef" }
      },
      "required": [ "key", "name", "equipment_a", "equipment_z",
"port_a", "port_z" ]
    },
    "port_relation": {
      "type": "object",
      "properties": {
        "class": { "type": "string", "enum":
[ "port_relation" ] },
        "parent": { "$ref": "#/$defs/keyRef" },
        "child": { "$ref": "#/$defs/keyRef" }
      },
      "required": [ "parent", "child" ]
    },
    "logical_connection_relation": {
      "type": "object",
      "properties": {
        "class": { "type": "string", "enum":
[ "logical_connection_relation" ] },
        "sequence": { "type": [ "integer", "null" ], "minimum":
1 },
        "route": { "type": [ "integer", "null" ], "minimum":
1 },
        "parent": { "$ref": "#/$defs/keyRef" },
        "child": { "$ref": "#/$defs/keyRef" }
      },
      "required": [ "parent", "child" ]
    },
    "network_topology": {
```

```
      "type": "object",
      "properties": {
        "class": { "type": "string", "enum":
 [ "network_topology" ] },
        "key": { "type": "string" },
        "name": { "type": "string" },
        "model_name": { "type": [ "string", "null" ] },
        "devices": { "type": "array", "items": { "key":
 "#/$defs/optionalKeyRef" } },
        "logical_connections": { "type": "array", "items":
 { "key": "#/$defs/optionalKeyRef" } }
      },
      "required": [ "key", "name", "devices",
 "logical_connections" ]
    }
  }
}
```

2. Decouple Connectivity from ETL

Ensure flexibility by separating device interaction logic from transformation logic (ETL ingestion). Users can develop their own collectors independently of the ETL logic.

Ensure the following points:

1. Design collectors to focus only on connectivity and data conversion to the unified schema.
2. Use or reuse any EMS/NMS adaptor (including third-party adaptors like Atrinet).
3. Push the standardized data into ServiceNow import sets.

3. Configure Generic ETL for CMDB Updates

Use a single reusable ETL to process all standardized import sets and update the CMDB accurately.

Ensure the following points:

1. Validate the import set contains data in the standardized JSON format.
2. Use the TSOM Generic ETL to:
   - Detect the entity type (e.g., Slot, Card, Logical Interface).
   - Map each entity to the correct CI class based on model mapping logic.
   - Populate fields including Inventory Category where applicable.

Special Handling:

- Logical Interfaces → Port table (`virtual=true`)
- Logical Connections → Logical Connection table
- PDU Cards → PDU table
- Equipment → Model-based CI class (most recent mapping if multiple exist)

4. Configure support for Multi-Chassis and Composite Devices

Model complex devices such as multi-chassis routers with correct CMDB relationships.

Ensure the following points:

1. Use the Logical Composite construct to represent grouped entities like Router + PDU.
2. Define individual components (Fan, Management, Slot) under their respective hierarchy.
3. Map each entity as per the TNI modeling guidance.

Example: For a 7750-2s multi-chassis device:

- Logical Composite → contains Router and PDU
- PDU → contains Slots → Cards → Sub-slots

5. Enable TNI Entity Creation Based on Installation

Ensure consistency with TNI standards without unnecessary record creation.

Ensure the following points:

- If TNI is installed:
  - Automatically create a TNI Entity for each discovered CI.
  - Set Inventory Category appropriately (e.g., "Logical Connection", "Interface").
- If TNI is not installed: Skip TNI entity creation to avoid orphaned or invalid records.

Configure service graph connectors

Configure Nokia Altiplano service graph connector

Configure the Nokia Altiplano service graph connector to import physical and logical inventory data from the Nokia Altiplano Access Network SDN Controller into your ServiceNow Configuration Management Database (CMDB).

This integration uses REST APIs (via a MID Server) to ensure the CMDB reflects accurate, up-to-date telecom inventory aligned with the TM Forum-based data model.

Note: A valid Telecommunications Service Operations Management subscription is required to use this connector.

Required plugins

Plugin	ID
Telecom Service Operations Core	`sn_tsom_core`
Service Graph Connector for Nokia Aliplano	sn_tsom_altiplano_connector

Note: External requirements:

- A running Nokia Altiplano Controller instance with access to its REST northbound API.
- A MID Server with secure connectivity to the Altiplano instance.

Configuration tasks overview

The following sections are available under the Nokia Altiplano navigation pane. Use the following table for post-guided setup or to perform manual configurations.

Section	Description
Setup	Configure MID Server, define Altiplano connections, and schedule imports.
Data Sources	Predefined data sources for bulk and filtered discovery (SGC-Nokia Altiplano Bulk Discovery, SGC-Nokia Altiplano Filtering Discovery). Enable parallel loading if needed. For more information on parallel loading, see Configure concurrent import and parallel loading.
Import Schedules	Manage scheduling for each Altiplano connection alias. Run jobs manually or at defined intervals.
Connections & Credential Aliases	Define aliases for each Altiplano instance. Store connection metadata and credentials.
Connections	Define Altiplano instance details, such as URL, selected MID Server, credential reference, and connection alias reference.
Credentials	Create Altiplano credentials using Basic Auth.
Filters	Configure filtering parameters used in filtered discovery (for example, by device IP or name).
Properties	Modify system behavior using connector-specific properties. For more information, see System components installed with Nokia Altiplano.

Supported Nokia Altiplano versions

- Minimum supported version: 24.6
- Supported OLT: Lightspan MF-2

Note: Fore more information, see Mapping Nokia Altiplano CIs and Relationships in CMDB.

Access the Guided Setup

Use the guided setup to simplify the configuration process. This setup provides an organized sequence of steps to help you complete integration quickly and correctly. To access the guided setup:

1. Navigate to **All** > **Service Graph Connectors** > **Nokia Altiplano** > **Setup**.
2. Follow the task sequence to configure MID Server settings, Nokia Altiplano connections, filters, and import schedules. For more information, see Set up Nokia Altiplano.

Related topics

Set up Nokia Altiplano

Set up multiple Nokia Altiplano instances

Configure concurrent import and parallel loading

Mapping Nokia Altiplano CIs and Relationships in CMDB

Run and verify a Nokia Altiplano import

System components installed with Nokia Altiplano

Telecom Discovery via Nokia Altiplano

Set up Nokia Altiplano

Learn how to install and configure the Service Graph Connector for Nokia Altiplano, including optional demo data, connectivity setup, and data collection schedules.

Before you begin

Role required: TSOM Visibility admin

About this task

Verify the active application scope is Service Graph Connector (SGC) for Nokia Altiplano.

1. Select the application picker ( ) in the header.
2. Search for **Nokia Altiplano**.
3. Select Service Graph Connector for Nokia Altiplano from the list.

Procedure

1. Navigate to **All** > **Service Graph Connectors** > **Nokia Altiplano** > **Setup**.
2. On the Getting started page, select **Get Started**.
   The Guided Setup home page opens in a new tab.
3. **Optional:** Create and configure MID Server or skip based on your environment.

   a. Select **Configure** to complete the process.

   b. Follow the on-screen instructions to download and install.

   c. Select **Mark as Complete** when done.

   > **Note:** For more information on how to install and configure MID Server, see Configuring MID Server.

   d. Configure or skip the validation step.

   e. Select **Mark as Complete** when validated.
   Once all MID Server steps are complete, proceed to **Configure Connectivity**.

4. Create connection aliases, credentials, and HTTP connections to your Nokia Altiplano instance with **Configure Connectivity**

   a. Select **Get Started**.

   b. Create and configure aliases for the connections and credentials:

      i. Select **Configure**.

      ii. In the **Name** field, specify the alias name.

      iii. Leave the rest of the fields as default, select **Submit** and then **Mark as Complete**.

      > **Note:** This enables using the connection by name rather than directly, enabling the collector to extract all active aliases from the CMDB and start performing data collection on the HTTP connection bound to it.

   c. Create the credentials to access the Nokia Altiplano Controller by selecting **Configure**.

i. In the **Name** field, specify the alias name.

ii. In the **User name** field, specify your Nokia Altiplano instance user name.

iii. In the **Password** field, specify your Nokia Altiplano instance password.

> **Note:** Other authentication fields may be required depending on the authentication methods used in your Nokia Altiplano instance. By default, we use Basic authentication credentials (as part of the Guided setup).

iv. Leave the rest of the fields as default, select **Submit** and then select **Mark as Complete**.

d. Create HTTP Connection by selecting **Configure**

i. In the **Name** field, specify the connection name.

ii. Choose the **Credentials** and **Connection Alias** created earlier.

iii. Specify the **Connection URL** for Nokia Altiplano.

iv. Check **Use MID Server** and choose either:

- Auto-select
- Specific MID Server
- Specific MID Cluster

v. Leave the rest of the fields as default, select **Submit** and then select **Mark as Complete**.

5. Configure Data Collection Schedule either for bulk discovery or filtered discovery by clicking **Get Started**and then click **Configure**.

To	Do the following
Schedule data collection for bulk discovery (Use bulk discovery data source for all de vices in connection alias)	○ Provide a **Name** for the scheduler. ○ Ensure the bulk data source SGC-Nokia Alti pano Bulk Discovery is selected. ○ Set **Active** the scheduler automatically runs as mentioned in the **Run** and **Time** fields. If inactive, you must manually exe cute. ○ Select **Use Connection.** ○ In the **Connection** field, select connection alias. For example, sn_sgc_altiplano.Ati plano_Alias. ○ In the **Run** field, select the frequency. Spec ify when you want this schedule to run: Dai ly, Weekly, Monthly, Periodically, After Par ent Runs, or Once. ○ In the **Time** field, enter the time in hours, minutes, and seconds.

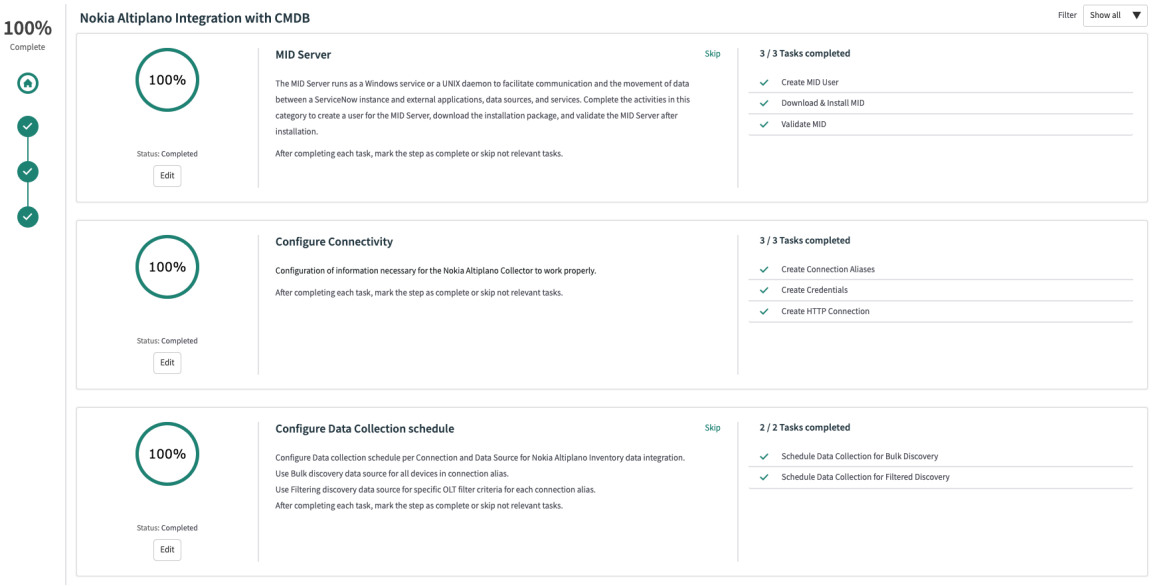
To	Do the following
Schedule data collection for filtered discovery (Use Filtering Discovery data source for specific OLT filter criteria for each connection alias)	**a.** Add Filtering Parameters **i.** In the **Connection Alias** field, choose a connection alias. **ii.** Optionally, in the **Filtered IPs** field, add IP filters in various formats defined as following: ▪ Single IP: `10.10.10.10` ▪ List: `10.10.10.10, 10.10.10.20` ▪ Ranges: `10.10.10.30-10.10.10.40` ▪ CIDR: `10.10.10.0/24` ▪ Mask: `10.10.10.0:255.255.255.0` **iii.** Optionally, add name filters to search by device name or part of the name. You can specify a single string or a list of names to filter the results. **b.** Schedule filtered Collection ▪ Provide a **Name** for the scheduler. ▪ Ensure the data source SGC-Nokia Altipano Filtering Discovery is selected. ▪ Set **Active** the scheduler automatically runs as mentioned in the **Run** and **Time** fields. If inactive, you must manually execute. ▪ In the **Run** field, select the frequency. Specify when you want this schedule to run: Daily, Weekly, Monthly, Periodically, After Parent Runs, or Once. ▪ In the **Time** field, enter the time in hours, minutes, and seconds. ▪ Select **Use Connection.**. ▪ In the **Connection** field, select connection alias. For example, sn_sgc_altiplano.Atiplano_Alias.

6. Select **Submit** and then select **Mark as Complete**.

Result

The configured connections, aliases, credentials, and import schedules can also be accessed from the navigation Navigate to **All** > **Service Graph Connectors** > **Nokia Altiplano**.

The following snapshot helps you confirm the configuration set up of Nokia Altiplano service graph connector.



Related topics

- Set up multiple Nokia Altiplano instances
- Configure concurrent import and parallel loading
- Mapping Nokia Altiplano CIs and Relationships in CMDB
- System components installed with Nokia Altiplano

Set up multiple Nokia Altiplano instances

Learn how to configure and manage multiple Nokia Altiplano instances within a single ServiceNow environment. This enables administrators and integrators to create independent connection aliases and import schedules customized to specific filtering or frequency needs.

Before you begin

Role required: TSOM Visibility admin

About this task

Confirm you have:

- You’re operating in the Service Graph Connector for NOKIA Altiplano application scope.
- Installed the Service Graph Connector for Nokia Altiplano.
- Completed the guided setup for the initial Altiplano instance.
- MID Server set up and validated.

You can configure additional Nokia Altiplano instances or reuse the same Altiplano instance with different connection aliases and import schedules. To add a new Altiplano instance, rerun the Guided setup to configure a new connection alias. Go through all connectivity stages in the setup for each new alias.

Procedure

1. Navigate to **All** > **Service Graph Connectors** > **Nokia Altiplano** > **Setup**.
2. On the Getting Started page, select **Get Started**.
   Repeat all the steps under the Configured Connectivity section. It creates configuration entries for the new instance of Nokia Altiplano. For more information, see Set up Nokia Altiplano.
3. Configure the connectivity by creating a connection alias, credentials, and HTTP connection.

a. In the Configure Connectivity section, select Get Started.

b. Select Configure to create a new connection alias by entering a unique alias name (for example, `Altiplano_Prod_01`).

c. Select **Submit** and mark the step as complete.

4. Configure credentials for the new Altiplano instance.

a. Specify a user name and password for the Altiplano Controller.

b. Submit and mark the step as complete.

5. Configure an HTTP connection.

a. Provide the connection name.

b. Select the newly created credentials and connection alias.

c. Enter the connection URL of the Nokia Altiplano instance.

d. Enable use MID Server and choose the appropriate MID option.

e. Select **Submit** and mark the step as complete.

6. Schedule data imports using either bulk or filtered discovery by configuring the import schedule.

a. Fill in the fields.
For more information, see Set up Nokia Altiplano.

b. In the **Use connection** field, choose the new Altiplano instance.

7. Select **Submit**.

Result

To confirm that your new instance setup is successful, you can verify the configuration.

- Navigate to All > Service Graph Connectors > Nokia Altiplano > Connection & Credential Aliases and confirm that the new alias is listed.
- Navigate to All > Service Graph Connectors > Nokia Altiplano > Import Schedule and confirm that the new import schedule is listed and confirm that corresponding jobs exist for each alias.

You can configure multiple connection aliases over the same Nokia Altiplano instance. This flexibility enables you to run imports at different frequencies and apply different filters to each alias.

Example: Altiplano_Weekly_OLT_10.10.10.*: Runs weekly, filters OLTs with IPs in the 10.10.10.* range. Altiplano_Daily_Prod: Runs daily, filters OLTs whose name contains "prod_olt".

Related topics

Configure concurrent import and parallel loading

Mapping Nokia Altiplano CIs and Relationships in CMDB

System components installed with Nokia Altiplano

Configure concurrent import and parallel loading

Improve the efficiency of large data imports from Nokia Altiplano by configuring concurrent imports and enabling parallel loading. This configuration allows the platform to run multiple data import and transformation jobs simultaneously.

Before you begin
Role required: TSOM Visibility admin

About this task
Concurrent import and parallel loading improve performance by dividing large datasets into smaller partitions, allowing multiple transformation and collection jobs to run in parallel. This setup is useful when importing large volumes of equipment, logical connections, and network topology data during bulk or filtered discoveries.

Use the following guidance to optimize performance:

- If transformation is slow: Enable Concurrent import in the import schedule.
- If data collection is slow: In addition to enabling concurrent import, configure Parallel loading in the data source and update the system property.

Note: Parallel loading works in addition to concurrent import. It does not replace it.

Procedure
1. Navigate to **All** > **Service Graph Connectors** > **Nokia Altiplano** > **Import Schedule**.
2. Configure concurrent import or concurrent import

To	Do the following
To enable concurrent import	**a.** Navigate to the **Import Schedule** for the dis covery job. **b.** Select the **Concurrent import** checkbox. **c.** Set the **Partition method** to **Custom size**. **d.** In the **Partition size**e field, enter the num ber of records per partition (for example, 1000). **Note:** The system splits the dataset into import sets based on the partition size. Each import set is processed in par allel, improving the speed of data trans formation.
Configure parallel loading in the data source	**a.** Navigate to the corresponding Data Source record. **b.** Select the **Parallel loading** checkbox. **c.** Navigate to Nokia Altiplano > Properties. **d.** Open the system property *sn_sgc_altiplano.parallel_number_of_data_* **e.** Set the value to the desired number of par allel jobs (for example, 3). **Note:** By default, the number of data source jobs is set to 1. Increasing this value allows the platform to execute multiple data collection jobs concur rently.

Related topics

- Set up Nokia Altiplano
- Mapping Nokia Altiplano CIs and Relationships in CMDB
- System components installed with Nokia Altiplano

Mapping Nokia Altiplano CIs and Relationships in CMDB

Use the Service Graph Connector for Nokia Altiplano to map discovered physical and logical network resources to telecom-aligned Configuration Item (CI) classes in the CMDB. The connector supports consistent service modeling, visibility into chassis-level components, and automation of logical and physical relationships.

To confirm accurate CI classification and insertion, the connector uses the Robust Transform Engine (RTE) and Identification and Reconciliation Engine (IRE).

The connector classifies and relates discovered CIs using telecom-specific models based on device type, function, and chassis structure. This helps maintain a clean and normalized CMDB across vendors. Discovered model names from Nokia Altiplano are automatically transformed into ServiceNow standard model identifiers and categories for slot and subslot components.

CI mapping and relationships

The following tables describe how Altiplano CIs are represented in the CMDB and how they relate to each other across physical and logical layers.

CMDB CI Mapping and Relationships (Physical Layer)

CI Type	CMDB Table	Description and Relationships
OLT CI	`cmdb_ci_optical_line_terminal`	Represents the OLT device. Contains Slot CIs and Logical Network Interface CIs.
ONU/ONT CI	`cmdb_ci_optical_network_terminal` or `cmdb_ci_optical_network_unit`	Represents ONU or ONT devices. The class is determined by the system property `sn_sgc_altiplano.onu_ci_class`. Contains Network Interface CIs.
Slot CI	`cmdb_ci_container_slot`	Represents main chassis slots. Contained by OLT CIs. Contains Interface Card CIs (for example, LT/NT, PSU, fan). Model transformations are applied in the data source. For more information, see the above mentioned Model transformation for slot and subslot CIs table.
Subslot CI	`cmdb_ci_container_subslot`	Represents subcomponents within interface cards (for example, cages for SFPs). Contained by LT/NT cards. Contains transceiver card CIs. For more information, see the above mentioned Model transformation for slot and subslot CIs table.
Interface Card CI	`cmdb_ci_interface_card`	Represents LT/NT cards, transceivers, and control units. Can contain subslots and network interfaces.

CMDB CI Mapping and Relationships (Physical Layer) (continued)

CI Type	CMDB Table	Description and Relationships
Network Interface CI	`cmdb_ci_ni_interface`	Represents both physical (for example, PON, Ethernet) and logical (for example, VLAN) ports. Contained by interface cards or ONU/ONT CIs. Logical ports are related to physical ports using Members::Member of.
Logical Connection CI	`cmdb_ci_ni_logical_path`	Represents logical paths such as PON or VLAN between OLT and ONU. Defined with Port A and Port Z attributes referencing terminating Network Interface CIs. VLAN paths consume PON paths.
IP Address CI	`cmdb_ci_ip_address`	Represents discovered IP addresses for OLTs. Owned by the corresponding OLT CI.

Key Relationship Examples

- Containment relationships
  - OLT CI → contains Slot CI
  - Slot CI → contains Interface Card CI
  - Interface Card CI → contains Subslot CI
  - Subslot CI → contains transceiver Interface Card CI
  - ONU/ONT CI → contains Network Interface CIs
- Interface relationships:
  - Logical Connections -> terminated by Network Interfaces
  - Network Interfaces -> members of Network Interfaces
- Logical path relationships:
  - VLAN path (parent) → consumes → PON path (child)
  - Logical paths → terminate at → Network Interface CIs via Port A and Port Z
- Ownership: IP Address CI → owned by OLT device

Supported models

1. Network equipment models (sn_ent_nw_equipment_model)
   - The supported OLT is Nokia Lightspan MF-2, by default the model name is "Nokia MF-2"
   - ONU/ONT models are manufacturer + ONU/ONT. the system property sn_sgc_altiplano.onu_ci_class defines if ONU or ONT will be used.
   - If the model wasn't found in the model table, a new model is created in the CI. and the CI will be created as "Network gear"
2. Equipment holder models: (sn_ent_nw_holder_model)
   - Slots models: "Traffic Slot", "FAN Slot", "Power Slot"
   - Subslots models: "SFP Subslot"

- The used model name can be customized by the customer through the Altiplano extension point (sn_sgc_altiplano.AltiplanoCustomizedModels)
- If the model wasn't found in the model table, a new model is created in the CI.

3. Network card models (sn_ent_nw_card_model)
   - Card models are found by the model name, manufacturer, and model number discovered from the Altiplano API
   - If the model wasn't found in the model table, a new model is created in the CI.
4. Network Interface Models: (sn_ent_nw_interface_model).
   - Ethernet ports models are found by the "port bandwidth" column in the Network Interface table (sn_ent_nw_interface_model). the port bandwidth of the port CI is located by the discovered port speed in the Bandwidth table (bandwidth)
   - PON physical ports models: "PON Access Interface", "PON Network Interfaces"
   - Logical ports models: "ENET Interface", "VLAN Interface", "LAG Interface", PON Logical Interface"
   - If the model wasn't found in the model table, the reference to the "Model ID" will remain empty.
5. Logical network connection models (sn_ent_logical_nw_connection_model)
   - PON Access Path
   - VLAN Path
   - If the model wasn't found in the model table, the reference to the "Model ID" will remain empty.

Note:

- If the connector cannot match a discovered equipment to an existing model in the product model table the CI is created as `Network Gear` by default.
- If demo data is installed, default models are created for OLT, ONU, ONT, Slots, Subslots, Cards, Network Interfaces, and Logical connections.
- Equipment and Equipment holder model names can be customized using an extension point

Model transformation for slot and subslot CIs

During ingestion, specific discovered model names are mapped to predefined CMDB model identifiers to confirm consistent slot categorization. The transformation logic is embedded in the SGC data source script and applies to the Nokia Altiplano source.

Slot components such as fan, power, and traffic slots are mapped to the `Slot` category in the CMDB. Subslot components such as SFP cages or synthetic subslots are mapped to the `Subslot` category in the CMDB.

Example: Nokia Altiplano slot and sub slot model mappings

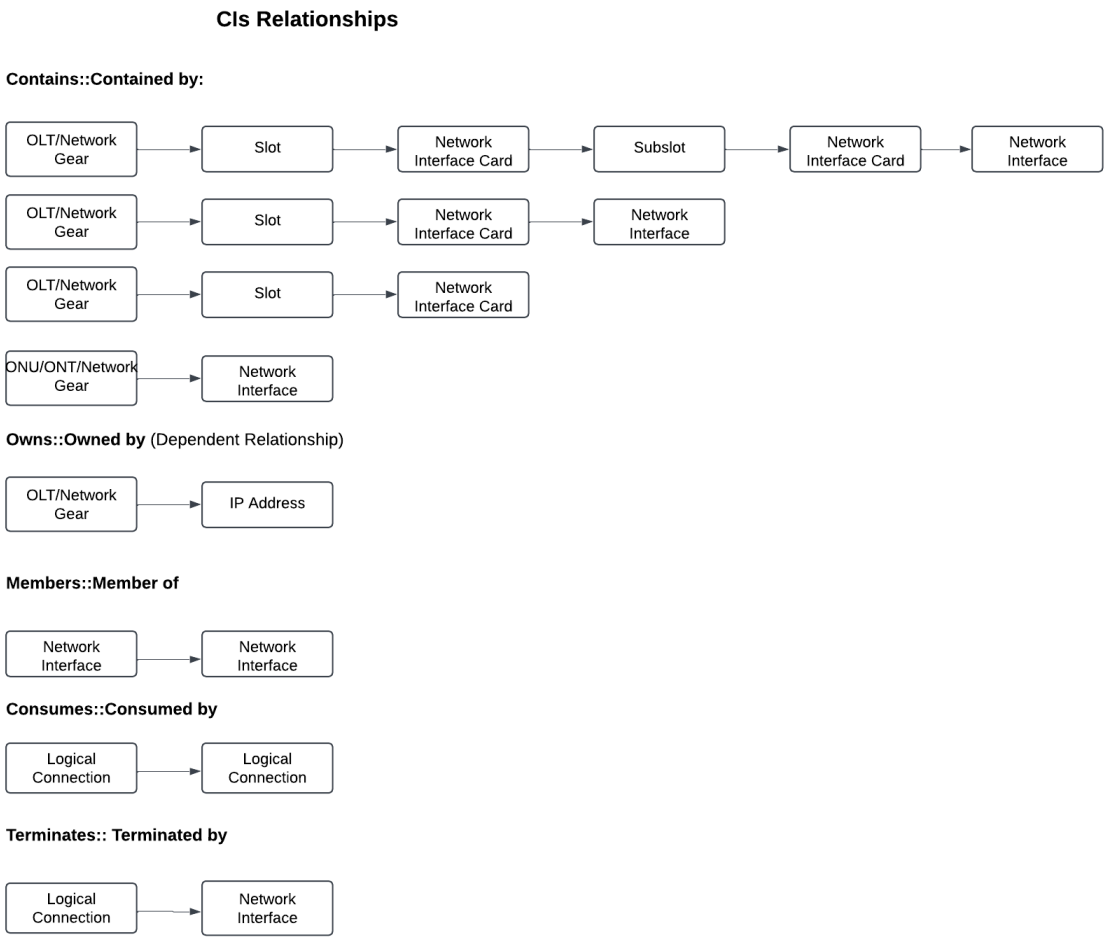
Source	Discovered Model Name	Target CMDB Model ID	Model Category
Altiplano	slot-fan	Fan Slot	Slot
Altiplano	slot-lt	Traffic Slot	Slot
Altiplano	cage	SFP Subslot	Subslot

Example: Nokia Altiplano slot and sub slot model mappings (continued)

Source	Discovered Model Name	Target CMDB Model ID	Model Category
Altiplano	slot-nt	Traffic Slot	Slot
Altiplano	synthetic nt-slot	SFP Subslot	Subslot
Altiplano	slot-psu	Power Slot	Slot

CI relationship structure

The following infographics describes the CI relationships.



Example: OLT to ONU Structure

The following structure enables end-to-end traceability from OLT through its hardware layers to the connected ONU and associated logical paths.

```
OLT (cmdb_ci_optical_line_terminal)
├── Slot
│   ├── Interface Card (LT/NT)
│   │   ├── Subslot
│   │   │   └── Interface Card (Transceiver)
│   │   │       └── Network Interface (PON port)
├── Logical Connections (PON/VLAN) → Connected to →
└── ONU/ONT (cmdb_ci_optical_network_terminal)
```

```
    └── Network Interface (ONT port)
```

Configure Cisco Meraki Service Graph Connector

Configure the Cisco Meraki Service Graph Connector (SGC) to import physical and logical inventory data from the Cisco Meraki (SGC) into your Configuration Management Database (CMDB).

Authentication

You must authenticate before initiating discovery. During the authentication process, the discovery service receives an access token, which it then uses for bulk or specific discovery operations. The integration uses Cisco Meraki REST APIs to discover managed elements such as network equipment, interfaces, and services.

This integration uses REST APIs (via a MID Server) to promote the CMDB reflects accurate, up-to-date telecom inventory aligned with the TM Forum-based data model. For a list of API references, see Cisco Meraki Service Graph Connector API Endpoints.

Note: A valid Telecommunications Service Operations Management subscription is required to use this connector.

Required plugins

Plugin	Plugin ID
Telecom Service Operations Core	sn_tsom_core
Service Graph Connector for Meraki Telco SD-WAN	sn_tsom_meraki_connector

Note: External requirements:

- A running Cisco Meraki instance with access to its REST API.
- A MID Server with secure connectivity to the Cisco Meraki instance.

Configuration tasks overview

The following sections are available under the Cisco Meraki navigation pane. Use the following table for post-guided setup or to perform manual configurations.

Section	Description
Setup	Configure MID Server, define Cisco Meraki connections, and schedule imports.
Data Sources	Predefined data sources for bulk and filtered discovery (SGC-Cisco Meraki Bulk Discovery, SGC-Cisco Meraki Filtering Discovery). Enable parallel loading if needed. For more information on parallel loading.
Import Schedules	Manage scheduling for each Cisco Meraki connection alias. Run jobs manually or at defined intervals.
Connections & Credential Aliases	Define aliases for each Cisco Meraki instance. Store connection metadata and credentials.
Connections	Define Cisco Meraki instance details, such as URL, selected MID Server, credential reference, and connection alias reference.

Section	Description
Credentials	Create Cisco Meraki credentials using Basic Auth.
Filters	Configure filtering parameters used in filtered discovery (for example, by device IP or name).
Properties	Modify system behavior using connector-specific properties. For more information, see.

Access the Guided Setup

Use the guided setup to simplify the configuration process. This setup provides an organized sequence of steps to help you complete integration quickly and correctly. For more information, see Set up the Service Graph Connector for Cisco Meraki schedule.

Set up the Service Graph Connector for Cisco Meraki schedule

Set up and configure the Service Graph Connector for Cisco Meraki. The process includes installation, optional demo data integration, connectivity configuration, and scheduling data collection for seamless Configuration Management Database (CMDB) integration.

Before you begin

Verify the active application scope is Service Graph Connector (SGC) for Cisco Meraki.

1. Select the application picker ( ) in the header.
2. Search for **Meraki**.
3. Select Service Graph Connector for Meraki from the list.

Role required: TSOM Visibility admin

Procedure

1. Navigate to **All** > **Service Graph Connectors** > **Meraki** > **Setup**.
2. On the Getting started page, select **Get Started**.
3. **Optional:** On the Guided setup home page, create and configure the MID Server or skip based on your environment.

   Note: For more information on how to activate and configure a MID Server, see Configuring MID Server.

   a. Select **Get Started**.

   b. Select **Configure**.

   c. Follow the on-screen instructions to download and install the appropriate MID Server installer archive for the operating system.

   d. Select **Mark as Complete** when done.
   Once all MID Server steps are complete, proceed to **Configure Connectivity**.

4. Enable the Service Graph Connector to reference the Meraki connection by name, extracting active aliases from the CMDB, and initiate data collection on the associated HTTP connection by creating connection aliases, credentials, and HTTP connections.

a. Select **Get Started**.

b. Create and configure aliases for the connections and credentials:

i. Select **Configure**.

ii. In the **Name** field, specify the alias name.

iii. Retain the default values in the rest of the fields.

iv. Select **Submit** and then select **Mark as Complete**.

c. Create the basic credentials to access the Meraki Cloud Controller.

i. Select **Configure**.

ii. In the **Name** field, specify the alias name.

iii. In the **API Key** field, enter the API key created in the dashboard.

> **Note:** Other authentication fields might be required depending on the authentication methods used in your Cisco Meraki instance. By default, use basic authentication credentials as part of the Guided setup. For more information, see Basic authentication credentials.

iv. Retain the default values in the rest of the fields.

v. Select **Submit** and then select **Mark as Complete**.

d. Create the HTTP Connection.

i. Select **Configure**.

ii. In the **Name** field, specify the connection name.

iii. Choose the **Credentials** and **Connection Alias** created earlier.

iv. Specify the **Connection URL** for Meraki.

v. Select the **Use MID Server** check box and indicate how the MID Server should be selected:

- Auto-select: Selects from MID Servers based on MID Server criteria, regardless of whether they are members of a cluster.
- Specific MID Server: Automatic reassignment to another MID Server is not supported.
- Specific MID Cluster: Automatic reassignment to another MID Server only selects from members of the specified cluster.

vi. Retain the default values in the rest of the fields.

vii. Select **Submit** and then select **Mark as Complete**.

5. Configure the data collection schedule.

a. Select **Get Started**.

b. Select **Configure**.

c. Configure the schedule.

i. In the **Name** field, provide a name for the scheduler.

ii. Select the data source based on whether the schedule should be for bulk discovery or filtered discovery.

- To use the bulk discovery data source for all devices in connection alias, select the data source SGC-Meraki Bulk Discovery.
- To set specific filter criteria for each connection alias, select the data source SGC-Meraki Filtering Discovery.

iii. Determine whether the schedule should automatically run or you want to have it run it manually.

iv.
- Select the **Active** check box, to run the schedule automatically based on the values in the **Run** and **Time** fields.
- Clear the**Active** check box to run the schedule manually.

v. From the **Run** drop-down list, select the frequency at which you want the scheduler to run automatically. The available choices are Daily, Weekly, Monthly, Periodically, after Parent Runs, or once.

vi. In the **Time** field, set the time of day at which the scheduler should run in hours, minutes, and seconds.

vii. Provide the connection information.

1. Select the **Use Connection** check box.
2. From the **Connection** drop-down list, select the connection alias. For example, sn_sgc_meraki.Meraki_Alias.

d. **Optional:** Set specific filter criteria for each connection alias.

i. Ensure that the [FIELD NAME FOR DATA SOURCE] field is set to the data source SGC-Meraki Filtering Discovery.

ii. In the **Connection Alias** field, choose a connection alias.

iii. Set the filtering parameters.

- To filter by IP address, in the **Filtered IPs** field, add IP filters in the desired format.
  - Single IP: `10.10.10.10`
  - List: `10.10.10.10, 10.10.10.20`
  - Ranges: `10.10.10.30-10.10.10.40`
  - CIDR: `10.10.10.0/24`
  - Mask: `10.10.10.0:255.255.255.0`
- To filter by name, add name filters to search by device name or part of the name.

  (Optional) You can specify a single string or a list of names.

6. Select **Submit** and then select **Mark as Complete**.

Result

Once the setup is complete, a confirmation screen appears indicating all tasks have been completed successfully.

You can access the configured connections, aliases, credentials, and import schedules by navigating to **All** > **Service Graph Connectors** > **Meraki**.

Manage multiple Cisco Meraki SD-WAN instances

Configure and manage multiple Cisco Meraki instances within a single ServiceNow AI Platform® environment. This functionality facilitates the creation of distinct connection aliases and the establishment of independent import schedules that you can customize to accommodate specific data filtering and frequency requirements for administrators and integrators.

Before you begin

Verify the following:

- The active application scope is Service Graph Connector (SGC) for Meraki.
- The SGC for Cisco Meraki has been installed.
- The initial Cisco Meraki instance has been set up. For more information, see Set up the Service Graph Connector for Cisco Meraki schedule.
- The associated MID Server has been set up and validated. For more information, see Configuring MID Server.

Role required: TSOM Visibility admin

About this task

You can configure additional Cisco Meraki instances or reuse the same Cisco Meraki instance with different connection aliases and import schedules. To add a new Cisco Meraki instance, run the guided setup to configure a new connection alias. All connectivity stages within the setup must be completed for each new alias.

Procedure

1. Navigate to **All** > **Service Graph Connectors** > **Meraki** > **Setup**.
2. On the Getting Started page, select **Get Started**.
3. Create and configure aliases for the connections and credentials:
   a. Select **Configure**.
   b. In the **Name** field, specify the alias name.
   c. Retain the default values in the rest of the fields.
   d. Select **Submit** and then select **Mark as Complete**.
4. Create the basic credentials to access the Meraki Cloud Controller.
   a. Select **Configure**.
   b. In the **API Key** field, enter the API key created in the dashboard.

      Note: Other authentication fields might be required depending on the authentication methods used in your Cisco Meraki instance. By default, use basic authentication credentials as part of the Guided setup. For more information, see Basic authentication credentials.

   c. In the **User name** field, specify your Cisco Meraki instance user name.
   d. Retain the default values in the rest of the fields.
   e. Select **Submit** and then select **Mark as Complete**.
5. Create the HTTP Connection.
   a. Select **Configure**.
   b. In the **Name** field, specify the connection name.

c. In the **Credentials** and **Connection Alias** fields, choose the items created earlier.

d. In the **Connection URL** field, select the URL for Cisco Meraki.

e. Select the **Use MID Server** check box and indicate how the MID Server should be selected:

- Auto-select: Automatically chooses the most appropriate MID Server.
- Specific MID Server: Select the name of the MID Server from the **MID Server** field that is displayed.
- Specific MID Cluster: Select the name of the MID cluster from the **MID Cluster** field that is displayed.

f. Retain the default values in the rest of the fields.

g. Select **Submit** and then select **Mark as Complete**.

6. Configure the connectivity by creating a connection alias, credentials, and HTTP connection.

a. In the Configure Connectivity section, select **Get Started**.

b. Select **Configure** to create a new connection alias by entering a unique alias name (for example, `Meraki_Prod_01`).

c. Select **Submit** and then select **Mark as Complete**.

7. Configure credentials for the new Meraki instance.

a. Specify a user name and password for the Meraki instance.

b. Select **Submit** and then select **Mark as Complete**.

8. Configure an HTTP connection.

a. Provide the connection name.

b. Select the newly created credentials and connection alias.

c. Enter the connection URL of the Meraki instance.

d. Enable use MID Server and choose the appropriate MID option.

e. Select **Submit** and then select **Mark as Complete**.

9. Schedule data imports using either bulk or filtered discovery by configuring the import schedule.

a. Select **Configure** and fill in the fields.
For more information, see Set up the Service Graph Connector for Cisco Meraki schedule.

b. In the **Use connection** field, choose the new Meraki instance.

10. Select **Submit**.

11. Confirm that your new instance setup is successful by verifying the configuration.

a. Navigate to > **All** > **Service Graph Connectors** > **Meraki** > **Connection & Credential Aliases**.

b. Confirm that the new alias is listed.
You can configure multiple connection aliases over the same Meraki instance. This flexibility enables you to run imports at different frequencies and apply different filters to each alias.

Configure Cisco Meraki allowlists

Configure connector definitions to limit polling to specific Cisco Meraki organizations.

Before you begin

Role required: tsom_visibility_admin

About this task

Use multiple connector instances with disjoint allowlists to distribute collection across MID Servers and avoid polling interval overruns in large-scale deployments. Only listed entities are included in the polling scope; all others are silently excluded.

Procedure

1. Navigate to **All** > **Event Management** > **Integrations** > **Connector Definitions**.
2. Search for `Meraki` in the search field.
3. Select the connector from the list, then the connector instance.
4. In the **organizationsToWhitelist** field in the **Connector Instance Values** section, enter the organization names to be included in the polling scope, separated by commas.
   For example, `CompanyG,Corp,SoutheastRegion`.
5. Select **Update**.

Result

The connector applies these values at each polling cycle.

Map Cisco Meraki CIs and relationships

Use the Service Graph Connector (SGC) for Cisco Meraki to map discovered physical and logical network resources to telecom-aligned configuration item (CI) classes in the Configuration Management Database (CMDB). Service Graph Connectors support consistent service modeling, provide visibility into chassis-level components, and automate the creation of logical and physical relationships.

To confirm accurate CI classification and insertion, the connector uses the Robust Transform Engine (RTE) and Identification and Reconciliation Engine (IRE).

The connector classifies and relates discovered CIs using telecom-specific models based on device type, function, and chassis structure. This organization helps maintain a clean and normalized CMDB across vendors. Discovered model names from Fortinet are automatically transformed into ServiceNow AI Platform standard model identifiers and categories for slot and subslot components.

CI mapping and relationships

The following table lists the CI object types in the CMDB that can be discovered, along with their representations in the CMDB and how they relate to one another.

CMDB CI Mapping and Relationships (Physical Layer)

CMDB CI Class	CMDB CI Table	CMDB Hierarchy	Object types / models	Description and Relationships
Network site	`cmdb_ci_ni_site`	CI → Site →Network Site	Organization network	• Represents the physical location of IP routers according to their longitude, latitude, and address.

CMDB CI Mapping and Relationships (Physical Layer) (continued)

CMDB CI Class	CMDB CI Table	CMDB Hierarchy	Object types / models	Description and Relationships
				• Network site contains IP routers and network interfaces. • Network site is a member of a group.
IP router	`cmdb_ci_ip_router`	CI → HW →NG → IP router	SD-WAN Edge / network or service router is represented by the IP router	• Represents the Meraki device. • Contains network interface CIs. • Contained by network sites and network service instances. • IP router is a member of a group.
Slot	`cmdb_ci_container_slot`	HW → Equipment → holder → Slot	Slot	• Slot is the main device in the network hierarchy. • Contains slots for IP routers, IP switches, power supply units, and fans.
Subslot	`cmdb_ci_container_subslot`	HW → Equipment → holder → Slot	Subslot	• Network card is the main device in

CMDB CI Mapping and Relationships (Physical Layer) (continued)

CMDB CI Class	CMDB CI Table	CMDB Hierarchy	Object types / models	Description and Relationships
				the network hierarchy. • Contains IP router, IP switch sublots (small form-factor pluggable or child cards).
Network Interface CI	`cmdb_ci_ni_interface`	Port → Network Port → Network interface	The list of support port models is defined in the vendor-specific network physical information.	• IP router, IP switch, or wireless access point is the main device in the network hierarchy. • Network card within the IP router or IP switch is the primary component. • Represents the physical ports contained within the device (IP router).
Network service instance	`cmdb_ci_network_service_instance`	CI → Service instance → Network service instance	Network service instance	• Network service instance includes IP routers and network interfaces. • Network site is a member of a group.

CMDB CI Mapping and Relationships (Physical Layer) (continued)

CMDB CI Class	CMDB CI Table	CMDB Hierarchy	Object types / models	Description and Relationships
Group	`cmdb_ci_group`	CI → Group	Represents organization	Network sites and network service instance are members.
IP Address CI	`cmdb_ci_ip_address`	CI → IP address	Represents discovered IP addresses for CIs.	Owned by the corresponding CI.

Configure a Fortinet SD-WAN Service Graph Connector

Configuring the Fortinet SD-WAN Service Graph Connector (SGC) enables you to import physical inventory data from FortiManager into the Configuration Management Database (CMDB) of your ServiceNow instance.

FortiManager is the Fortinet centralized management platform that enables you to configure, monitor, and manage multiple Fortinet security devices, including firewalls and SD-WAN appliances, from a single interface.

The FortiManager JSON API is used to perform configuration and monitoring tasks on a FortiManager device. The SGC for Fortinet SD-WAN uses JSON to gather information and populate the CMDB.

By using structured JSON requests over HTTPS, you can efficiently interact with FortiManager to streamline operations and scale network management tasks across multiple Fortinet devices. These APIs enable automated tasks within the Fortinet ecosystem, such as bulk configuration changes, device management, status monitoring, and inventory collection. To access the Fortinet APIs, create a user and key credentials from FortiManager. For API reference examples, see Fortinet Service Graph Connector API Endpoints.

Note: A valid Telecommunications Service Operations Management subscription is required to use this connector.

Required plugins

Plugin	Plugin ID
Telecom Service Operations Core	sn_tsom_core
Service Graph Connector for Fortinet Telco SD-WAN	sn_tsom_fortinet_connector

Note: External requirements:

- A running FortiManager instance with access to its JSON-RPC based northbound API.
- A configured API key in the FortiPortal to enable access to JSON-RPC requests.
- A MID Server with secure connectivity to the FortiManager instance.

Configuration tasks overview

The following sections are available under the Fortinet navigation pane. Use the following table for post-guided setup or to perform manual configurations.

Section	Description
Setup	Configure the MID Server, define Fortinet connections, and schedule imports.
Data Sources	Predefined data sources for bulk (SGC-Fortinet Bulk Discovery) and filtered discovery (SGC-Fortinet Filtering Discovery).
Import Schedules	Manage scheduling for each Fortinet connection alias. Run jobs manually or at defined intervals.
Connections & Credential Aliases	Define aliases for each Fortinet instance. Store connection metadata and credentials.
Connections	Define Fortinet instance details, such as the URL, the selected MID Server, credential reference, and connection alias reference.
Credentials	Create Fortinet credentials using API Key Credentials.
Filters	Configure filtering parameters used in discovery filtered by device IP or name.
Properties	Modify system behavior using connector-specific properties.

Access the Guided Setup

Use the guided setup to simplify the configuration process. This setup provides an organized sequence of steps to help you complete integration quickly and correctly. For more information, see Set up the Service Graph Connector for Fortinet schedule.

Set up the Service Graph Connector for Fortinet schedule

Set up and configure the Service Graph Connector for Fortinet SD-WAN. The process includes installation, optional demo data integration, connectivity configuration, and scheduling data collection for seamless Configuration Management Database (CMDB) integration.

Before you begin

Verify the active application scope is Service Graph Connector (SGC) for Fortinet.

1. Select the application picker () in the header.
2. Search for **Fortinet**.
3. Select Service Graph Connector for Fortinet from the list.

Role required: TSOM Visibility admin

Procedure

1. Navigate to **All** > **Service Graph Connectors** > **Fortinet** > **Setup**.
2. On the Getting started page, select **Get Started**.
3. **Optional:** On the Guided setup home page, create and configure the MID Server or skip based on your environment.

Note: For more information on how to activate and configure a MID Server, see Configuring MID Server.

a. Select **Get Started**.

b. Select **Configure**.

c. Follow the on-screen instructions to download and install the appropriate MID Server installer archive for the operating system.

d. Select **Mark as Complete** when done.
Once all MID Server steps are complete, proceed to **Configure Connectivity**.

4. Enable the Service Graph Connector to reference the Fortinet connection by name, extracting active aliases from the CMDB, and initiate data collection on the associated HTTP connection by creating connection aliases, credentials, and HTTP connections.

a. Select **Get Started**.

b. Create and configure aliases for the connections and credentials:

i. Select **Configure**.

ii. In the **Name** field, specify the alias name.

iii. Retain the default values in the rest of the fields.

iv. Select **Submit** and then select **Mark as Complete**.

c. Create the basic credentials to access the FortiManager.

i. Select **Configure**.

ii. In the **Name** field, specify the alias name.

iii. In the **API Key** field, enter the API key created in the FortiPortal.

Note: Other authentication fields might be required depending on the authentication methods used in your FortiManager instance. By default, use basic authentication credentials as part of the Guided setup. For more information, see Basic authentication credentials.

iv. Retain the default values in the rest of the fields.

v. Select **Submit** and then select **Mark as Complete**.

d. Create the HTTP Connection.

i. Select **Configure**.

ii. In the **Name** field, specify the connection name.

iii. Choose the **Credentials** and **Connection Alias** created earlier.

iv. Specify the **Connection URL** for FortiManager.

v. Select the **Use MID Server** check box and indicate how the MID Server should be selected:

- Auto-select: Selects from MID Servers based on MID Server criteria, regardless of whether they are members of a cluster.
- Specific MID Server: Automatic reassignment to another MID Server is not supported.
- Specific MID Cluster: Automatic reassignment to another MID Server only selects from members of the specified cluster.

vi. If **Specific MID Server** or **Specific MID Cluster** is selected, select a name in the appropriate field.

vii. Retain the default values in the rest of the fields.

viii. Select **Submit** and then select **Mark as Complete**.

5. Configure the data collection schedule.

a. Select **Get Started**.

b. Select **Configure**.

c. Configure the schedule.

i. In the **Name** field, provide a name for the scheduler.

ii. Select the data source based on whether the schedule should be for bulk discovery or filtered discovery.

- To use the bulk discovery data source for all devices in connection alias, select the data source SGC-Fortinet Bulk Discovery.
- To set specific filter criteria for each connection alias, select the data source SGC-Fortinet Filtering Discovery.

iii. Determine whether the schedule should automatically run or you want to have it run it manually.

iv.
- Select the **Active** check box, to run the schedule automatically based on the values in the **Run** and **Time** fields.
- Clear the**Active** check box to run the schedule manually.

v. From the **Run** drop-down list, select the frequency at which you want the scheduler to run automatically. The available choices are Daily, Weekly, Monthly, Periodically, after Parent Runs, or once.

vi. In the **Time** field, set the time of day at which the scheduler should run in hours, minutes, and seconds.

vii. Provide the connection information.

1. Select the **Use Connection** check box.
2. From the **Connection** drop-down list, select the connection alias. For example, sn_sgc_fortinet.Fortinet_Alias.

d. Optional: Set specific filter criteria for each connection alias.

i. Ensure that the [FIELD NAME FOR DATA SOURCE] field is set to the data source SGC-Fortinet Filtering Discovery.

ii. In the **Connection Alias** field, choose a connection alias.

iii. Set the filtering parameters.

- To filter by IP address, in the **Filtered IPs** field, add IP filters in the desired format.
  - Single IP: `10.10.10.10`
  - List: `10.10.10.10, 10.10.10.20`
  - Ranges: `10.10.10.30-10.10.10.40`
  - CIDR: `10.10.10.0/24`
  - Mask: `10.10.10.0:255.255.255.0`
- To filter by name, add name filters to search by device name or part of the name.

  (Optional) You can specify a single string or a list of names.

6. Select **Submit** and then select **Mark as Complete**.

Result

Once the setup is complete, a confirmation screen appears indicating all tasks have been completed successfully.

You can access the configured connections, aliases, credentials, and import schedules from the navigation. Navigate to **All** > **Service Graph Connectors** > **Fortinet**.

Manage multiple Fortinet SD-WAN instances

Configure and manage multiple Fortinet instances within a single ServiceNow AI Platform® environment. This functionality facilitates the creation of distinct connection aliases and the establishment of independent import schedules that you can customize to accommodate specific data filtering and frequency requirements for administrators and integrators.

Before you begin

Verify the following:

- The active application scope is Service Graph Connector (SGC) for Fortinet.
- The SGC for Fortinet has been installed.
- The initial Fortinet instance has been set up. For more information, see Set up the Service Graph Connector for Fortinet schedule.
- The associated MID Server has been set up and validated. For more information, see Configuring MID Server.

Role required: TSOM Visibility admin

About this task

You can configure additional Fortinet instances or reuse the same Fortinet instance with different connection aliases and import schedules. To add a new Fortinet instance, run the guided setup to configure a new connection alias. All connectivity stages within the setup must be completed for each new alias.

Procedure

1. Navigate to **All** > **Service Graph Connectors** > **Fortinet** > **Setup**.
2. On the Getting Started page, select **Get Started**.
3. Create and configure aliases for the connections and credentials:
   a. Select **Configure**.
   b. In the **Name** field, specify the alias name.

c. Retain the default values in the rest of the fields.

d. Select **Submit** and then select **Mark as Complete**.

4. Create the basic credentials to access the FortiManager.

a. Select **Configure**.

b. In the **API Key** field, enter the API key created in the FortiPortal.

> **Note:** Other authentication fields might be required depending on the authentication methods used in your Fortinet instance. By default, use basic authentication credentials as part of the Guided setup. For more information, see Basic authentication credentials.

c. In the **User name** field, specify your FortiManager instance user name.

d. Retain the default values in the rest of the fields.

e. Select **Submit** and then select **Mark as Complete**.

5. Create the HTTP Connection.

a. Select **Configure**.

b. In the **Name** field, specify the connection name.

c. In the **Credentials** and **Connection Alias** fields, choose the items created earlier.

d. In the **Connection URL** field, select the URL for Fortinet.

e. Select the **Use MID Server** check box and indicate how the MID Server should be selected:

- Auto-select: Automatically chooses the most appropriate MID Server.
- Specific MID Server: Select the name of the MID Server from the **MID Server** field that is displayed.
- Specific MID Cluster: Select the name of the MID cluster from the **MID Cluster** field that is displayed.

f. Retain the default values in the rest of the fields.

g. Select **Submit** and then select **Mark as Complete**.

6. Configure the connectivity by creating a connection alias, credentials, and HTTP connection.

a. In the Configure Connectivity section, select **Get Started**.

b. Select **Configure** to create a new connection alias by entering a unique alias name (for example, `Meraki_Prod_01`).

c. Select **Submit** and then select **Mark as Complete**.

7. Configure credentials for the new Fortinet instance.

a. Specify a user name and password for the Fortinet instance.

b. Select **Submit** and then select **Mark as Complete**.

8. Configure an HTTP connection.

a. Provide the connection name.

b. Select the newly created credentials and connection alias.

c. Enter the connection URL of the Fortinet instance.

d. Enable use MID Server and choose the appropriate MID option.

e. Select **Submit** and then select **Mark as Complete**.

9. Schedule data imports using either bulk or filtered discovery by configuring the import schedule.

   a. Select **Configure** and fill in the fields.
   For more information, see Set up the Service Graph Connector for Fortinet schedule.

   b. In the **Use connection** field, choose the new Fortinet instance.

10. Select **Submit**.
11. Confirm that your new instance setup is successful by verifying the configuration.

    a. Navigate to > **All** > **Service Graph Connectors** > **Fortinet** > **Connection & Credential Aliases**.

    b. Confirm that the new alias is listed.
    You can configure multiple connection aliases over the same Fortinet instance. This flexibility enables you to run imports at different frequencies and apply different filters to each alias.

Configure Fortinet allowlist

Configure connector definitions to limit polling to specific Fortinet ADOMs using allowlist settings.

Before you begin
Role required: tsom_visibility_admin

About this task
Use multiple connector instances with disjoint allowlists to distribute collection across MID Servers and avoid polling interval overruns in large-scale deployments. Specify the ADOM from which the connector pulls data.

Procedure

1. Navigate to **All** > **Event Management** > **Integrations** > **Connector Definitions**.
2. Search for `Fortinet` in the search field.
3. Select the connector from the list, then the Connector Instance.
4. In the **Connector Instance Values** section, specify the ADOM from which you want to receive metric data.
   The connector retrieves logs only for the specified ADOM.
5. Select **Update**.

Result
The connector applies these values at each polling cycle.

Filter ADOMs for the Fortinet pull connector

Additional filter conditions required in Fortinet pull connector.

Before you begin
Role required: `tsom_visibility_admin`

About this task
By default, the Fortinet pull connector collects metric data from all ADOMs. To limit collection to ADOMs that meet specific conditions, configure the following parameters on the connector instance (for example, **Fortinet SD-WAN SLA Log**):

ADOM filter parameters

Parameter	Default	Description
`adomFilterEnabled`	`false`	Turns ADOM filtering on or off. When `false`, the connector collects all ADOMs and the filter is bypassed. When `true`, the connector collects only the ADOMs that the filter class accepts.
`adomFilterClass`	`FortinetAdomFilterVF`	Name of the MID Server script include that implements the filter. Evaluated only when `adomFilterEnabled` is `true`. The class must be active, deployed to the MID Server, and expose a `shouldIncludeAdom(adomData)` handler that returns a Boolean.

Two filter classes are provided:

- `FortinetAdomFilterVF` (default) collects only ADOMs where the `restricted_prds` property equals `fabric` and the `VF SDWAN Service Provided` meta field equals `yes`.
- `FortinetAdomFilter` collects all ADOMs unconditionally.

> **Warning:** Filtering fails closed. If you set `adomFilterEnabled` to `true` and `adomFilterClass` is not a valid script include that is deployed to the MID Server, all ADOMs are blocked and metric collection stops. Enable filtering only when the named class populates the conditions your FortiManager deployment supports.

Procedure

1. Navigate to **All** > **Event Management** > **Connectors**.
2. Open the Fortinet pull connector instance.
3. On the **Additional Parameters** related list, set `adomFilterEnabled` to `true`.
4. Set `adomFilterClass` to the name of the filter class you want to apply.
   Use `FortinetAdomFilterVF` for the Vodafone filter, `FortinetAdomFilter` to collect all ADOMs, or the name of a custom filter class.
5. Save the connector instance.
   The connector applies the filter at the next polling cycle and collects metric data only from the ADOMs that the filter class accepts.

What to do next

To apply your own filtering logic, create a MID Server script include (`ecc_agent_script_include`) that exposes a `shouldIncludeAdom(adomData)` handler returning `true` to collect the ADOM or `false` to skip it, deploy it to the MID Server, and set `adomFilterClass` to its name.

Filter ADOM discovery for the Fortinet Service Graph Connector

Restrict which Fortinet ADOMs the SGC discovers by activating the predefined Vodafone filter or by providing your own implementation of the ADOM filter extension point.

Before you begin

Role required: `tsom_visibility_admin`

About this task

By default, the SGC for Fortinet discovers all ADOMs returned by FortiManager.
To restrict discovery to a subset of ADOMs based on their properties, use the `sn_sgc_fortinet.FortinetCustomizedAdomFilter` extension point. An implementation defines a `shouldIncludeAdom(adom)` handler that returns `true` to discover the ADOM or `false` to skip it, where `adom` is the ADOM object returned by the Fortinet API.

A predefined implementation, `FortinetDefaultAdomFilter`, is provided for Vodafone deployments. It discovers only ADOMs where the **VF SDWAN Service Provided** meta field equals `yes` and the *`restricted_prds`* property equals `fabric`. This implementation is inactive by default, so it doesn't affect discovery unless you activate it.

> ⚠ **Warning:** Activate the predefined filter only if your FortiManager deployment populates the required meta fields. If you activate it without them, no ADOM matches the filter and discovery returns no data.

Procedure

1. Navigate to **All** > **System Definition** > **Extension Instances**.
2. Filter the list by **Point** equals `sn_sgc_fortinet.FortinetCustomizedAdomFilter`.
3. Open the **FortinetDefaultAdomFilter** record.
4. Set **Active** to true and save the record.
   The SGC applies the filter at the next polling cycle and discovers only the ADOMs that satisfy it.

What to do next

To apply your own filtering logic instead, create an implementation of the `sn_sgc_fortinet.FortinetCustomizedAdomFilter` extension point in your own scope, define the `shouldIncludeAdom(adom)` handler, and activate its extension instance. Leave the predefined `FortinetDefaultAdomFilter` instance inactive.

Customize scripted extension points

Customize how license expiration dates for Fortinet devices are stored on CIs by implementing scripted extension points.

Before you begin

Service Graph Connector for Fortinet must be installed. For instructions, see Configure a Fortinet SD-WAN Service Graph Connector.

Role required: tsom_visibility_admin

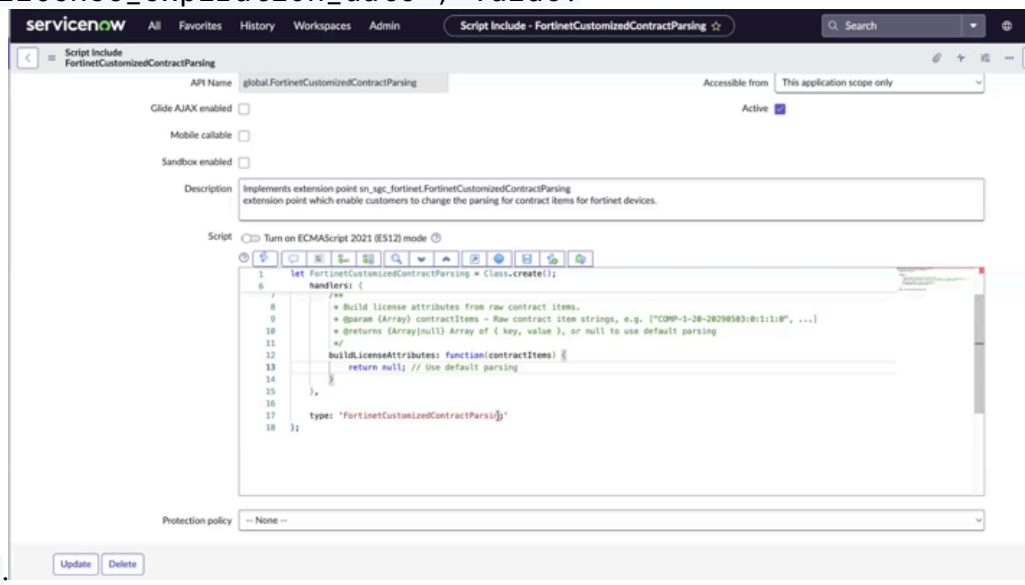
About this task

By default, Fortinet SGC stores license expiration dates as separate CI key-value pairs for each device, using the naming convention `license_expiration_date_`*`SERVICE_CODE`*. You can customize this behavior using scripted extension points to implement alternative storage logic, such as storing only the earliest expiration date across all devices.

Procedure

1. Navigate to **All** > **System Scripted Extension Points** > **Scripted Extension Points**.
2. Search for Fortinet in the **Extension Points** search field.
3. From the **API Name** search results list, select **sn_gnc_fortinet.FortinetCustomizedContractParsing**.
4. Select the **Create implementation** related link.

5. In the **Script** field, modify the `buildLicenseAttributes` function to return an array of `{key, value}` objects containing the CI key-value pairs you want to store.
   For example, to store only the earliest expiration date across all devices instead of one key-value pair per device, the script would need to loop through `contractItems`, find the minimum date value, and return it as a single object: `{ key: "license_expiration_date", value: "<earliest_date>" }`.



6. Run your implementation before the default by setting the **Order** field to a value less than `100`.
   The default implementation has an order of 100; implementations with a lower order number execute first and take precedence.
7. Select **Update**.

Result

Your custom implementation is saved and active. The next time Fortinet SGCSGC discovers devices, your script runs first and stores the license key-value pairs you defined on the relevant CIs, overriding the default per-device key-value pairs behavior.

Related topics

Extension points

Create a scripted extension point

Map Fortinet CIs and relationships

Use the Service Graph Connector (SGC) for Fortinet SD-WAN to map discovered physical and logical network resources to telecom-aligned configuration item (CI) classes in the Configuration Management Database (CMDB). Service Graph Connectors support consistent service modeling, provide visibility into chassis-level components, and automate the creation of logical and physical relationships.

To confirm accurate CI classification and insertion, the connector uses the Robust Transform Engine (RTE) and Identification and Reconciliation Engine (IRE).

The connector classifies and relates discovered CIs using telecom-specific models based on device type, function, and chassis structure. This organization helps maintain a clean and normalized CMDB across vendors. Discovered model names from Fortinet are automatically transformed into ServiceNow AI Platform standard model identifiers and categories for slot and subslot components.

CI mapping and relationships

The following table lists the CI object types in the CMDB that can be discovered, along with their representations in the CMDB and how they relate to one another.

CMDB CI Mapping and Relationships (Physical Layer)

CMDB CI Class	CMDB CI Table	CMDB Hierarchy	Object types/ models	Description and Relationships
Network site	`cmdb_ci_ni_site`	CI → Site →Network Site	Organization network	• Represents the physical location of IP routers according to their longitude, latitude, and address. • Network site contains IP routers and network interfaces. • Network site is a member of a group.
IP router	`cmdb_ci_ip_router`	CI → HW →NG → IP router	SD-WAN Edge/ network or service router is represented by the IP router	• Represents the FortiGate device. • Contains network interface CIs. • Contained by network sites and network service instances. • IP router is a member of a group.
Slot	`cmdb_ci_container_slot`	HW → Equipment → holder → Slot	Slot	• Slot is the main device in the network hierarchy.

CMDB CI Mapping and Relationships (Physical Layer) (continued)

CMDB CI Class	CMDB CI Table	CMDB Hierarchy	Object types/ models	Description and Relationships
				• Contains slots for IP routers, IP switches, power supply units, and fans.
Subslot	`cmdb_ci_container_subslot`	HW → Equipment → holder → Slot	Subslot	• Network card is the main device in the network hierarchy. • Contains IP router, IP switch sublots (small form-factor pluggable or child cards).
Network Interface CI	`cmdb_ci_ni_interface`	Port → Network Port → Network interface	The list of support port models is defined in the vendor-specific network physical information.	• IP router, IP switch, or wireless access point is the main device in the network hierarchy. • Network card within the IP router or IP switch is the primary component. • Represents the physical ports contained

CMDB CI Mapping and Relationships (Physical Layer) (continued)

CMDB CI Class	CMDB CI Table	CMDB Hierarchy	Object types/ models	Description and Relationships
				within the device (IP router).
Network service instance	`cmdb_ci_network_service_instance`	CI → Service instance → Network service instance	Network service instance	• Network service instance includes IP routers and network interfaces. • Network site is a member of a group.
Group	`cmdb_ci_group`	CI → Group	Represents organization	Network sites and network service instance are members.
IP Address CI	`cmdb_ci_ip_address`	CI → IP address	Represents discovered IP addresses for CIs.	Owned by the corresponding CI.

Discovered logical interfaces

In addition to physical ports, the SGC discovers logical interfaces, such as tunnel and VPN interfaces (for example, `Hub1 - inetVPN`, `inetVPN`, and `mplsVPN`), from the Fortinet inventory. Each logical interface is created as a network interface CI (`cmdb_ci_ni_interface`) and related to its parent device (IP router) CI.

Because a CMDB CI exists for each logical interface, metrics that report against a logical interface name are mapped to the corresponding logical interface CI rather than to the parent device CI. This mapping enables interface-level metric correlation and reporting for tunnel and VPN interfaces.

Firmware version calculation

When the SGC discovers an IP router, it sets the firmware version on the `cmdb_ci_ip_router` CI by joining the device `os_ver`, `mr`, and `patch` fields from the Fortinet inventory with periods. For example, a device with `os_ver=7`, `mr=4`, and `patch=11` resolves to `7.4.11`, matching the value shown in the Fortinet GUI. If any of these fields is missing, the firmware version is left empty.

To calculate the firmware version differently, override the default with the `sn_sgc_fortinet.FortinetCustomizedFirmwareVersion` extension point. Create an implementation that defines a `formatFirmwareVersion(device)` handler, where

`device` is the Fortinet device object returned by the API. Return your own version string from the handler, or return `null` to use the default calculation.

Configure Arista VeloCloud Service Graph Connector

Configure the Arista VeloCloud Service Graph Connector (SGC) to discover and import physical and logical inventory data from your Arista VeloCloud environment into the Configuration Management Database (CMDB).

Arista VeloCloud relies on VeloCloud Orchestrator (VCO) as its centralized management platform, available as either a self-hosted or SaaS deployment, to configure, monitor, and manage SD-WAN inventory across the network.

Authentication

VeloCloud Orchestrator supports two types of user roles: MSP admins and organization admins. To authenticate API requests, users must generate an API token from their VCO account.

There are two types of API tokens:

- User token: Inherits the same access permissions as the user who created it.
- Organization token: Provides access scoped to a single organization.

You must authenticate before initiating discovery. During the authentication process, the discovery service receives an access token, which it then uses for bulk or specific discovery operations. The integration uses VeloCloud REST APIs to discover managed elements such as network equipment, interfaces, and services.
This integration uses REST APIs (via a MID Server) to ensure the CMDB reflects accurate, up-to-date telecom inventory aligned with the TM Forum-based data model. For a list of API references, see .

Note: A valid Telecommunications Service Operations Management subscription is required to use this connector.

Required plugins

Plugin	Plugin ID
Telecom Service Operations Core	sn_tsom_core
Service Graph Connector for Velocloud Telco SD-WAN	sn_tsom_vcloud_connector

Note: External requirements:

- A running Arista VeloCloud instance with access to its REST API.
- A MID Server with secure connectivity to the Arista VeloCloud instance.

Configuration tasks overview

The following sections are available under the Arista VeloCloud navigation pane. Use the following table for post-guided setup or to perform manual configurations.

Section	Description
Setup	Configure MID Server, define Arista VeloCloud connections, and schedule imports.

Section	Description
Data Sources	Predefined data sources for bulk and filtered discovery (SGC-Arista VeloCloud Bulk Discovery, SGC-Arista VeloCloud Filtering Discovery). Enable parallel loading if needed. For more information on parallel loading.
Import Schedules	Manage scheduling for each Arista VeloCloud connection alias. Run jobs manually or at defined intervals.
Connections & Credential Aliases	Define aliases for each Arista VeloCloud instance. Store connection metadata and credentials.
Connections	Define Arista VeloCloud instance details, such as URL, selected MID Server, credential reference, and connection alias reference.
Credentials	Create Arista VeloCloud credentials using Basic Auth.
Filters	Configure filtering parameters used in filtered discovery (for example, by device IP or name).
Properties	Modify system behavior using connector-specific properties. For more information, see.

Access the Guided Setup

Use the guided setup to simplify the configuration process. This setup provides an organized sequence of steps to help you complete integration quickly and correctly. For more information, see Set up the Service Graph Connector for Arista VeloCloud schedule.

Set up the Service Graph Connector for Arista VeloCloud schedule

Set up and configure the Service Graph Connector for Arista VeloCloud SD-WAN. The process includes installation, optional demo data integration, connectivity configuration, and scheduling data collection for seamless Configuration Management Database (CMDB) integration.

Before you begin

Verify the active application scope is Service Graph Connector (SGC) for VeloCloud.

1. Select the application picker ( ) in the header.
2. Search for **VeloCloud**.
3. Select Service Graph Connector for VeloCloud from the list.

Role required: TSOM Visibility admin

Procedure

1. Navigate to **All** > **Service Graph Connectors** > **VeloCloud** > **Setup**.
2. On the Getting started page, select **Get Started**.
3. **Optional:** On the Guided setup home page, create and configure the MID Server or skip based on your environment.

   Note: For more information on how to activate and configure a MID Server, see Configuring MID Server.

   a. Select **Get Started**.

   b. Select **Configure**.

c. Follow the on-screen instructions to download and install the appropriate MID Server installer archive for the operating system.

d. Select **Mark as Complete** when done.
Once all MID Server steps are complete, proceed to **Configure Connectivity**.

4. Enable the Service Graph Connector to reference the VeloCloud connection by name, extracting active aliases from the CMDB, and initiate data collection on the associated HTTP connection by creating connection aliases, credentials, and HTTP connections.

a. Select **Get Started**.

b. Create and configure aliases for the connections and credentials:

i. Select **Configure**.

ii. In the **Name** field, specify the alias name.

iii. Retain the default values in the rest of the fields.

iv. Select **Submit** and then select **Mark as Complete**.

c. Create the basic credentials to access the VeloCloud Orchestrator.

i. Select **Configure**.

ii. In the **Name** field, specify the alias name.

iii. In the **API Key** field, enter the API key created in the VeloCloud Orchestrator.

> **Note:** Other authentication fields might be required depending on the authentication methods used in your instance. By default, use basic authentication credentials as part of the Guided setup. For more information, see Basic authentication credentials.

iv. Retain the default values in the rest of the fields.

v. Select **Submit** and then select **Mark as Complete**.

d. Create the HTTP Connection.

i. Select **Configure**.

ii. In the **Name** field, specify the connection name.

iii. Choose the **Credentials** and **Connection Alias** created earlier.

iv. Specify the **Connection URL** for VeloCloud Orchestrator.

v. Select the **Use MID Server** check box and indicate how the MID Server should be selected:

- Auto-Select: Selects from MID Servers based on MID Server criteria, regardless of whether they are members of a cluster.
- Specific MID Server: Automatic reassignment to another MID Server is not supported.
- Specific MID Cluster: Automatic reassignment to another MID Server only selects from members of the specified cluster.

vi. If **Specific MID Server** or **Specific MID Cluster** is selected, select a name in the appropriate field.

vii. Retain the default values in the rest of the fields.

viii. Select **Submit** and then select **Mark as Complete**.

5. Configure the data collection schedule.

a. Select **Get Started**.

b. Select **Configure**.

c. Configure the schedule.

i. In the **Name** field, provide a name for the scheduler.

ii. Select the data source based on whether the schedule should be for bulk discovery or filtered discovery.

- To use the bulk discovery data source for all devices in connection alias, select the data source VeloCloud Full Discovery.
- To set specific filter criteria for each connection alias, select the data source VeloCloud Filtered Discovery.

iii. Determine whether the schedule should automatically run or you want to have it run it manually.

iv.
- Select the **Active** check box, to run the schedule automatically based on the values in the **Run** and **Time** fields.
- Clear the**Active** check box to run the schedule manually.

v. From the **Run** drop-down list, select the frequency at which you want the scheduler to run automatically. The available choices are Daily, Weekly, Monthly, Periodically, After Parent Runs, or Once.

vi. In the **Time** field, set the time of day at which the scheduler should run in hours, minutes, and seconds.

vii. Provide the connection information.

1. Select the **Use Connection** check box.
2. From the **Connection** drop-down list, select the connection alias. For example, sn_sgc_vcloud.Velocloud_Alias.

d. **Optional:** Set specific filter criteria for each connection alias.

i. Ensure that the [FIELD NAME FOR DATA SOURCE] field is set to the data source VeloCloud Filtered Discovery.

ii. In the **Connection Alias** field, choose a connection alias.

iii. Set the filtering parameters.

- To filter by IP address, in the **Filtered IPs** field, add IP filters in the desired format.
  - Single IP: `10.10.10.10`
  - List: `10.10.10.10, 10.10.10.20`
  - Ranges: `10.10.10.30-10.10.10.40`
  - CIDR: `10.10.10.0/24`
  - Mask: `10.10.10.0:255.255.255.0`

- To filter by name, add name filters to search by device name or part of the name.

  (Optional) You can specify a single string or a list of names.

6. Select **Submit** and then select **Mark as Complete**.

Result
Once the setup is complete, a confirmation screen appears indicating all tasks have been completed successfully.

You can access the configured connections, aliases, credentials, and import schedules from the navigation. Navigate to **All** > **Service Graph Connectors** > **VeloCloud**.

Manage multiple Arista VeloCloud instances

Configure and manage multiple Arista VeloCloud instances within a single ServiceNow AI Platform® environment. This functionality facilitates the creation of distinct connection aliases and the establishment of independent import schedules that you can customize to accommodate specific data filtering and frequency requirements for administrators and integrators.

Before you begin
Verify the following:

- The active application scope is Service Graph Connector (SGC) for VeloCloud.
- The SGC for Arista VeloCloud has been installed.
- The initial Arista VeloCloud instance has been set up. For more information, see Set up the Service Graph Connector for Arista VeloCloud schedule.
- The associated MID Server has been set up and validated. For more information, see Configuring MID Server.

Role required: TSOM Visibility admin

About this task
You can configure additional VeloCloud instances or reuse the same VeloCloud instance with different connection aliases and import schedules. To add a new VeloCloud instance, run the guided setup to configure a new connection alias. All connectivity stages within the setup must be completed for each new alias.

Procedure
1. Navigate to **All** > **Service Graph Connectors** > **VeloCloud** > **Setup**.
2. On the Getting Started page, select **Get Started**.
3. Create and configure aliases for the connections and credentials:
   a. Select **Configure**.
   b. In the **Name** field, specify the alias name.
   c. Retain the default values in the rest of the fields.
   d. Select **Submit** and then select **Mark as Complete**.
4. Create the basic credentials to access the VeloCloud Orchestrator.
   a. Select **Get Started**.
   b. Select **Configure**.

c. In the **API Key** field, enter the API key created in the Orchestrator.

> **Note:** Other authentication fields might be required depending on the authentication methods used in your Arista VeloCloud instance. By default, use basic authentication credentials as part of the Guided setup. For more information, see Basic authentication credentials.

d. In the **User name** field, specify your Arista VeloCloud instance user name.

e. Retain the default values in the rest of the fields.

f. Select **Submit** and then select **Mark as Complete**.

5. Create the HTTP Connection.
   a. Select **Get Started**.
   b. Select **Configure**.
   c. In the **Name** field, specify the connection name.
   d. In the **Credentials** and **Connection Alias** fields, choose the items created earlier.
   e. In the **Connection URL** field, select the URL for Cisco Meraki.
   f. Select the **Use MID Server** check box and indicate how the MID Server should be selected:
      - Auto-select: Automatically chooses the most appropriate MID Server.
      - Specific MID Server: Select the name of the MID Server from the **MID Server** field that is displayed.
      - Specific MID Cluster: Select the name of the MID cluster from the **MID Cluster** field that is displayed.
   g. Retain the default values in the rest of the fields.
   h. Select **Submit** and then select **Mark as Complete**.
6. Configure the connectivity by creating a connection alias, credentials, and HTTP connection.
   a. In the Configure Connectivity section, select **Get Started**.
   b. Select **Configure** to create a new connection alias by entering a unique alias name (for example, `Velocloud_Prod_01`).
   c. Select **Submit** and then select **Mark as Complete**.
7. Configure credentials for the new VeloCloud instance.
   a. Specify a user name and password for the VeloCloud instance.
   b. Select **Submit** and then select **Mark as Complete**.
8. Configure an HTTP connection.
   a. Provide the connection name.
   b. Select the newly created credentials and connection alias.
   c. Enter the connection URL of the VeloCloud instance.
   d. Enable use MID Server and choose the appropriate MID option.
   e. Select **Submit** and then select **Mark as Complete**.
9. Schedule data imports using either bulk or filtered discovery by configuring the import schedule.

a. Select **Configure** and fill in the fields.
For more information, see Set up the Service Graph Connector for Arista VeloCloud schedule.

b. In the **Use connection** field, choose the new VeloCloud instance.

10. Select **Submit**.
11. Confirm that your new instance setup is successful by verifying the configuration.
    a. Navigate to > **All** > **Service Graph Connectors** > **VeloCloud** > **Connection & Credential Aliases**.
    b. Confirm that the new alias is listed.
    You can configure multiple connection aliases over the same VeloCloud instance. This flexibility enables you to run imports at different frequencies and apply different filters to each alias.

Map Arista VeloCloud CIs and relationships

Use the Service Graph Connector (SGC) for Arista VeloCloud to map discovered physical and logical SD-WAN resources to telecom-aligned configuration item (CI) classes in the Configuration Management Database (CMDB). SGC support consistent service modeling, provide visibility into network components, and automate the creation of logical and physical relationships.

To confirm accurate CI classification and insertion, the connector uses the Robust Transform Engine (RTE) and Identification and Reconciliation Engine (IRE).

The connector classifies and relates discovered CIs using telecom-specific models based on device type, function, and chassis structure. This organization helps maintain a clean and normalized CMDB across vendors. Discovered model names from Fortinet are automatically transformed into ServiceNow AI Platform standard model identifiers and categories for slot and subslot components.

CI mapping and relationships

The following table lists the CI object types in the CMDB that can be discovered, along with their representations in the CMDB and how they relate to one another.

CMDB CI Mapping and Relationships (Physical Layer)

CMDB CI Class	CMDB CI Table	CMDB Hierarchy	Object types / models	Description and Relationships
Network site	`cmdb_ci_ni_site`	CI → Site →Network Site	Organization network	• Represents the physical location of IP routers according to their longitude, latitude, and address. • Network site contains IP routers

CMDB CI Mapping and Relationships (Physical Layer) (continued)

CMDB CI Class	CMDB CI Table	CMDB Hierarchy	Object types / models	Description and Relationships
				and network interfaces. • Network site is a member of a group.
IP router	`cmdb_ci_ip_router`	CI → HW →NG → IP router	SD-WAN Edge / network or service router is represented by the IP router	• Represents the VeloCloud device. • Contains network interface CIs. • Contained by network sites and network service instances. • IP router is a member of a group.
Network Interface CI	`cmdb_ci_ni_interface`	Port → Network Port → Network interface	The list of support port models is defined in the vendor-specific network physical information.	• IP router or wireless access point is the main device in the network hierarchy. • Network card within the IP router is the primary component. • Represents the physical ports contained within the device (IP router).

CMDB CI Mapping and Relationships (Physical Layer) (continued)

CMDB CI Class	CMDB CI Table	CMDB Hierarchy	Object types / models	Description and Relationships
Network service instance	`cmdb_ci_network_service_instance`	CI → Service instance → Network service instance	Network service instance	• Network service instance includes IP routers and network interfaces. • Network site is a member of a group.
Group	`cmdb_ci_group`	CI → Group	Represents organization	Network sites and network service instance are members.
IP Address CI	`cmdb_ci_ip_address`	CI → IP address	Represents discovered IP addresses for CIs.	Owned by the corresponding CI.

Configure Telecom Assurance

Setting up and configuring fault management enables you to monitor network devices for configuration and performance issues.

Configure a webhook to monitor SD-WAN network devices using Event Management. This application collects event data from your device and generates alerts.

All events are received in the ServiceNow AI Platform® dashboard and automatically mapped to alerts. Event rules evaluate each incoming event and determine whether to create an alert or link it to an existing one. You can define custom event rules, receive notifications using webhook mechanism, and integrate with external systems through the Integrations Launchpad.

Configuring Telecommunications API notifications

Configure the Telecommunications API notification in the ServiceNow instance.

Modeling the Telecommunications API notification workflow

The following steps help to configure the Telecommunications API notification in the ServiceNow instance.

1. Create a topic: You can create topics either by manually typing the external message details or automatically collecting the available topics from the external system.
2. Create a topic subscription: You subscribe to the available topics for incoming notifications from the external system, based on the customer preference. Additionally, you generate the callback URL and register the subscription.
3. Activate the Telecommunications Alarm Management Open API endpoint: To receive responses from the external system, activate the subscribed endpoints of the Telecommunications Alarm Management Open API connection in the Workflow Studio.
4. Provide the callback URL to the external system for receiving notifications. Customer can also reuse the callback URL. When requests from TMF 688 hit the Callback URL, it initiates the *`Default Alarm Event Notification Trigger`* flow to create an event.

   To learn more about the functions to handle Event Notification Management Open API requests that are triggered by external trigger definitions to create, update, and delete events, see Event Notification Management Open API and TMFTopicEventAPIUtilOOB - Scoped.

This workflow creates an event in the Event Management application. To learn more about using Event Management, see Event Management.

Related topics

System components installed with Telecommunications API notifications

External event management via Telecommunications API notifications

Create a topic

Create a topic and publish the incoming notifications from the external system to the topic. By creating the topics, subscribers can select the topics to which they want to subscribe.

Before you begin

Make sure that the Telecommunications Alarm Management Open API (sn_ind_tmf642) application is installed with the ServiceNow AI Platform.

Role required: admin, sn_api_notif_ mgmt.topic_creator

About this task

You can create topics either by manually typing the external message details or automatically collecting the available topics from the external system. When you create a topic, it creates a record in the Topic [sn_api_notif_mgmt_topic] table.

Procedure

1. **All** > **Telecom API Notification** > **Topics**.
2. Select **New**.
   If you've integrated with an external system, you can select **Get Topics** to get the available topics automatically. This action triggers the *`Event Alarm Notification API`* subflow. To learn more about the functions that enable you to query and manipulate records in the topic, see TopicUtilOOB - Scoped.
3. On the form, fill in the fields.

Topic form

Field	Description
Topic id	Unique topic id.

Field	Description
Topic name	Name of the topic.
Type	Type of topic. Select one from the following: ○ Ingress: Option for inbound notification. ○ Egress: Option for outbound notification.
Header query	Encoded header query parameters. To learn more about the query parameters that follow the TMF 688 standards, see the TM Forum.
Content query	Encoded content query parameters. To learn more about the query parameters that follow the TMF 688 standards, see TM Forum.
Description	A brief description about the topic.

4. Select **Submit**.

Result
A topic is created.

What to do next
You can create the topic subscription according to the customer requirement. For more information, see .

Create a topic subscription

Subscribe to the topic in the ServiceNow AI Platform that you want respond to the incoming notification from the external system. By subscribing to the topic, the subscriber receives the notifications based on the topics that you subscribe to.

Before you begin
- Make sure that the Telecommunications Alarm Management Open API (sn_ind_tmf642) application is installed with the ServiceNow AI Platform.
- Create topics for the incoming notifications.

Role required: admin, sn_api_notif_mgmt.subscription_creator

About this task
You subscribe to the available topics for the incoming notifications from the external system, based on the customer preference. You generate the callback URL to share with the customers. When a request from an external system hits the callback URL, it initiates the creation of an event in the Event Management application.

Additionally, you register the topic subscription to start receiving the incoming notifications. When you create a topic subscription, it creates a record in the Topic Subscription [sn_api_notif_mgmt_subscription] table. To learn more about the methods to query and manipulate records in the Topic Subscription, see TopicSubscriptionUtilOOB - Scoped.

Procedure

1. **All** > **Telecom API Notification** > **Subscription**.
2. Select **New**.
3. On the form, fill in the fields.

Topics Subscription form

Field	Description
Topic	Topic that you want to subscribe.
CallbackURL	The callback URL that you're sharing with the external system to capture the incoming notification. The URL is generated automatically when you select **Generate CallbackURL**.
Filter query	Encoded content query parameters from the topic. You can also modify the filter query. To learn more about the query parameters that follow the TMF 688 standards, see TM Forum.
Registration status	Status of the Topic registration with the external system. By default, it's **Unregistered**. If the process is successful, the field value changes to **Registered**. Otherwise it's **Error**.
Registration message	Registration status message from the external system.
Subscription id	Unique subscription id from the external system.

4. Get the callback URL by selecting **Generate CallbackURL**.
5. Register the subscription by selecting **Register**.

Result

A trigger definition is created for the callback URL and the topic is registered to the external system.

What to do next

In the Workflow Studio, you activate the endpoints of the Telecommunications Alarm Management Open API connection. For more information, see Activate the Telecommunications Alarm Management Open API endpoint.

Activate the Telecommunications Alarm Management Open API endpoint

Activate the endpoint of the Telecommunications Alarm Management Open API connection. By activating the endpoint, you receive the incoming notifications from the external system for the topic that you registered.

Before you begin

- Create the topic and subscribe to it to receive the incoming notifications.
- Generate a callback URL and register the topic subscription.

Role required: admin

About this task

You activate the subscribed endpoints of the Telecommunications Alarm Management Open API connection in the Workflow Studio to receive responses from the external system.

Procedure

1. Navigate to **All** > **Process Automation** > **Flow Designer**.
2. Select the **Integrations** tab, then select **Connections**.
3. Switch the toggle to **Inbound**.
4. Select **Telecommunications Alarm Management Open API**.
5. Open the endpoint record that you want to activate.
6. Select **Activate**.

Configure event and metric connectors

Configure a webhook

Integrate with a webhook to connect to an external event source and push event information to your ServiceNow instance.

Before you begin

The plugins listed in the following table must be installed.

Plugin	Plugin ID	Plugin description
TSOM Event Management Connectors	sn_tsom_em_connectors	TSOM Assurance Connectors for events and metrics
TSOM Event Management Core	sn_tsom_em_core	TSOM Assurance Core features

Role required: TSOM Assurance admin

Procedure

1. Navigate to > **Workspaces** > **Service Operations Workspace**.
2. From the navigation pane, select the Integrations Launchpad icon  .
3. Select the **Browse Integrations** tab
4. Search for the desired integration (for example, Meraki or Fortinet).
5. Select the integration tile labeled **Events**.
6. On the Provide details page, provide details about the connector type.
   a. In the **Connector name** field, enter a unique name for the connector type.
   b. **Optional:** In the **Tags** field, enter tags to help locate and identify connectors of this type on the Express List.
   c. **Optional:** Add additional tags.

i. Select the plus sign next to the **Tags** field.

ii. In the Add tags window, provide the tag key and tag value.

iii. Select **Add**.

d. **Optional:** In the **Description** field, enter information to help identify this connector type.

e. Verify that the vendor name in the **Source** field is correct.

f. **Optional:** From the **Assignment group** drop-down list, select the group or team that is responsible for managing and maintaining the Push connector.

g. Select **Next**.

7. On the Set-up push connector page, copy the auto-generated URL parameter value to the clipboard by selecting **Copy to clipboard** in the **URL parameter value** field.

   This URL is automatically generated.

8. Follow the instructions on the page to integrate your third-party connector with Event Management.
9. Select **Activate**.
10. In the confirmation message, select **OK**.
11. **Optional:** In the **Details** tab, view the connector details.
    The connector is saved and a notification is sent after the connector is active.
12. **Optional:** After you receive a notification that the connector is active, confirm the successful configuration of the push connector and the flow of events into the system by selecting the **Events** tab.
    Incoming events might take a few minutes to appear in the dashboard.

Related topics

Exploring Event Management

Cisco Meraki installed integrations

Fortinet installed integrations

Configure an event pull connector

Configure event pull connectors that require a script, connector definition, and connector instance to pull events from external management systems. These connectors automate the data retrieval process, promoting the seamless integration of external events into your system for efficient monitoring and management.

Before you begin

Role required: TSOM Assurance admin

Procedure

1. Navigate to **Workspaces** > **Service Operations Workspace**.
2. From the navigation pane, select the Integrations Launchpad icon  .
3. Select the **Browse Integrations** tab

4. Search for the external management system integration (for example, Meraki, Fortinet, or VeloCloud).
5. Select the integration tile labeled **Metrics**.
6. On the Connector Instance New record page, in the **Name** field, enter a unique name for the connector type.
7. In the **Connector definition** field, select the type of integration you're setting up.
8. **Optional:** In the **Description** field, enter brief information about this connector.
9. **Optional:** In the **Assignment group** field, select the group or team responsible for managing and maintaining the connector.
10. In the **Host IP** field, enter the IP address or host name used to select the appropriate MID Server for communicating with the event source host.
11. In the **Credential** field, select the valid credentials to access the event source host.
12. **Optional:** Validate the connectivity of the connector before activating it by selecting **Test Connector**.
13. Select **Update**.
14. **Optional:** Verify that the pull connector is configured correctly and events are flowing into the system by returning to the Integrations Launchpad.
    The tiles appear under the **Installed Integrations** tab.

Related topics

Exploring Metric Intelligence

Cisco Meraki installed integrations

Fortinet installed integrations

Configure elastic event pull connectors for MPN

Configure a mobile private network (MPN) pull connector instance to collect metrics from specified KPI domains in Elastic data store and publish them for assurance monitoring and analytics.

Before you begin

The MID Server MID Server must have Access Control (ACL) records for each of the following tables:

- `sn_tsom_em_conns_kpi_definitions` (KPI definitions table)
- `cmdb_ci_network_service_instance`
- `cmdb_ci_app`

For more information, see Set up a MID Server role.

Roles required: tsom_assurance_admin and assigned roles for MID Server.

Procedure

1. Navigate to **All** > **Event Management** > **Integrations** > **Connector Definitions**.
2. Select **Nokia MPN Pull Connector**.
3. Open an existing connector instance or create one:

- To use an existing instance, select it from the **Connector Instances** list and edit only the fields you need to change.
- To create an instance, select **New** and complete all required fields as described in the following steps.

4. On the form, fill in the fields.

Field	Description
Host IP	URL of the MPN host
Credential	Username and password for basic authentication
kpi_domain identifier	Domains to collect metrics from. Leaving the field empty collects metrics from all domains. Entering one or more domain identifiers, with values separated by commas (for example, `RAN, Core`) collect metrics only from specific domains.
Metrics collection schedule	How long, in seconds, each metrics collection should run.

5. Add a MID Server.
   a. In the **MID Servers for Connectors** section, select the plus icon next to **Insert a new row**.
   b. In the **MID Server** field, enter the name of the MID Server.
   c. Save your selection by selecting the green check mark.
6. Enable the connector instance by selecting the **Active** check box.
   The connector begins collecting metrics from Elastic data store.
7. Select **Update**.

Related topics

View metric to CI and resource binding

Event field mapping configuration

Create an event rule to bind metric events to host CIs

View metric values in the Insights Explorer

Create an Insights Explorer view

Configure elastic connectors for MPN alarm collection

Configure a connector instance to collect fault management alarm data from a Mobile Private Network (MPN) Elastic index and forward events to Event Management.

Before you begin

1. Create Basic Auth Credentials
2. Create HTTP(S) Connection
3. Activate the Telecommunications Alarm Management Open API endpoint

Role required: tsom_assurance_admin

About this task

The MPN connector collects alarm and event data from managed network infrastructure by querying Elastic fault management indices at a configured polling interval. Each connector instance collects from one or more of these indices, each corresponding to a specific network domain. The MPN pull connector definition supports three connector instances: one for alarm collection and two for metrics collection. This procedure covers configuring the alarm collection instance.

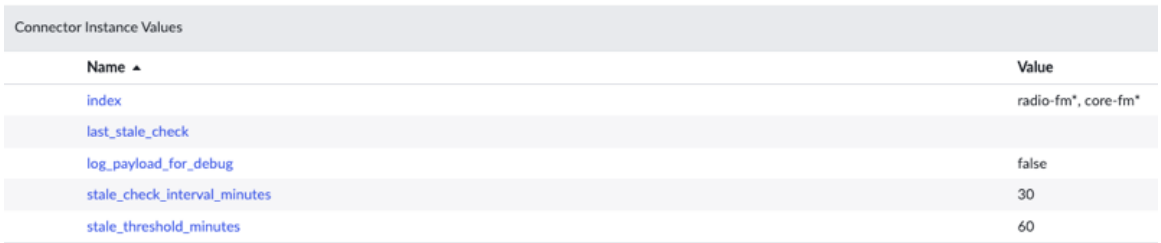
You can create multiple connector instances, one per index, and assign each to a dedicated MID Server to distribute the polling workload across your deployment. A default index value is provided for each connector instance.

The alarm collection instance also clears stale alarms: alarms that were cleared at the source but were missed during a normal collection cycle and would otherwise remain open in Event Management. At a regular interval, the connector rechecks the source for active alarms that have not been updated within a set time window and clears any that are no longer active. Default values are provided, so no action is required unless you want to tune this behavior.

Procedure

1. Navigate to **All** > **Event Management** > **Integrations** > **Connector Definitions**.
2. Select **Nokia MPN Pull Connector**.
3. In the **Default schedule** field, enter the polling time (in seconds) to be used for all connector instances.
   The default value is 120 seconds.
4. In the Connector Instances related list, select **Nokia MPN Alarms**.
5. On the Connector Instances page, in the **Event collection schedule** field, enter how long each event collection should run, in seconds.
   This value overrides the default schedule set on the connector definition. The MPN Alarms instance defaults to 60 seconds.
6. In the **Index** field, provide the index connector instance values.

   Enter the index pattern for the network domain this connector instance collects from. To collect from more than one domain, enter the patterns as a comma-separated list. The following table lists the standard index patterns:


Connector Instance Values

Name ▲	Value
index	radio-fm*, core-fm*
last_stale_check	
log_payload_for_debug	false
stale_check_interval_minutes	30
stale_threshold_minutes	60

Index connector instance values

Description	Name
Radio network elements	radio-fm*
Core network elements	core-fm*
Application-layer network elements	application-fm*
IXR (Interconnect Router) devices	ixr-fm*
DAC (Digital Access Controller) devices	dac-fm*

7. Add the MID Server.
   a. In the **MID Servers for Connectors** section, select the plus icon next to **Insert a new row**.
   b. Enter the name of the MID Server in the **MID Server** field.
   c. Select the green check mark to save your selection.
8. Enable the connector instance by selecting the **Active** check box.
9. **Optional:** Tune how stale alarms are detected and cleared.
   Set the following connector instance values on the alarm collection instance. If you don't set them, the default values apply.

Stale alarm detection parameters

Parameter	Default	Description
`stale_threshold_minutes`	60	How long, in minutes, an active alarm can go without an update at the source before the connector treats it as stale and clears it in Event Management.
`stale_check_interval_minutes`	30	How often, in minutes, the connector runs the stale alarm check. This check runs independently of the event collection schedule.

10. Select **Update**.
11. **Optional:** View your configured alarms by navigating to **em_event.LIST** and searching for `API Alarm Notification`.

Override default metric-to-CI binding

Replace the shipped logic that binds collected metrics to configuration items (CIs) for a Telecommunications Service Operations Management metric source by creating your own implementation of the `EventFieldMapping` extension point and wiring it into an event field mapping rule.

Before you begin

- Set your application scope to **Telecommunications Service Operations Management Event Management Connectors** (`sn_tsom_em_conns`) using the scope switcher. If you work in the Global scope, the implementation you create in step 1 is generated in the wrong scope and the resolver can't find it at run time.
- Identify the source name as it appears on the metric and on the corresponding event. The source name is the value used to resolve which scripted extension handles the event.
- Decide which rule name your custom implementation will own. Each implementation is identified by a unique combination of source name and rule name. You will use this same rule name in the script include's `getRuleName()` method and in the rule's **Name** field, and the two must match exactly.
- Determine which CMDB class and field your implementation should match against, and which event fields contain the values to match.

Role required: `tsom_assurance_admin`

For an overview of how metric-to-CI binding works in Telecommunications Service Operations Management service graph connectors, see Metric-to-CI binding.

About this task

When a connector collects metric data, the metric framework stores it and creates a metric-to-CI mapping record. Because the record does not yet have a configuration item (CI), the framework automatically generates an event for it. That event triggers the event field mapping rules. A rule whose source matches the event runs a script that resolves the CI and sets the `cmdb_ci` field on the event, and the resolved CI is then reflected in the CI column of the Metric To CI Mappings table:

metric-to-CI mapping record → event created → event field mapping rule → `cmdb_ci` set on the event → configuration item shown in the Metric To CI Mappings table

The shipped behavior is provided by an `EventFieldMapping` implementation that the rule's script retrieves through a resolver. The resolver, `EventFieldMappingResolver.getByRule()`, returns the implementation whose `getVendor()` matches the source name you pass and whose `getRuleName()` matches the rule name. That implementation exposes a `mapCi()` method that performs the CI lookup. To override the shipped behavior for a connector source, create your own implementation under the same source name with a new rule name, set its `getVendor()`, `getRuleName()`, and `mapCi()` methods, and wire it into an event field mapping rule whose script delegates to the resolver.

Important: The resolver matches on names, and a mismatch fails silently: no error is thrown, the CI is simply never set. The source name returned by `getVendor()` must match the literal you pass to `getByRule()` in the rule script, and the rule name returned by `getRuleName()` must match the rule's **Name**. A typo, a case difference, or the wrong application scope all produce the same result: the resolver returns nothing and the CI is not set. The resolver itself doesn't log the mismatch; the "No EventFieldMapping implementation found" message is written by the fallback in the rule script you create in step 3.

Procedure

1. Create an implementation of the `sn_tsom_em_conns.EventFieldMapping` extension point.
   a. Navigate to **All** > **System Extension Points** > **Scripted Extension Points**.
   b. Search for the API name `sn_tsom_em_conns.EventFieldMapping` and open it.
   c. In the **Related Links** section, select **Create implementation**.
   The platform generates a script include and registers it as an extension instance automatically, so the resolver can find it. Creating the implementation from this link is what registers the instance; you don't register it as a separate step.
   d. Set the script include name (for example, `NokiaMPNMappingRule`) and save it.
2. Implement the extension point methods in the generated script include.
   a. In `getVendor()`, return your source name (for example, `return 'Nokia MPN';`).
   The resolver matches this value, so it must be exact.
   b. In `getRuleName()`, return your rule name (for example, `return 'NokiaMPNMappingRule';`).
   This value must match the **Name** of the rule you create in step 3.

c. In `mapCi(eventGr, origEventSysId, fieldMappingRuleName)`, read the values you need from the event (for example, fields under `additional_info`), look up the matching configuration item, and set the `cmdb_ci` and `ci_type` fields on the event.

d. Return `true` from `mapCi()` to continue event processing, or `false` to drop the event.

e. Save the script include.

Note: For example, the shipped Nokia MPN implementation resolves the CI on the table named by `additional_info.ciClass` (default `cmdb_ci`), trying these event fields in order: `additional_info.name` (direct name match), `additional_info.pmDataSource.dn` (bottom-up DN traversal, deepest matching component wins), `serial_no`, `hw_id`, and `nhg_alias`. On a match, it sets `cmdb_ci` and `ci_type` on the event.

3. Create an event field mapping rule that delegates to your implementation.

a. Navigate to **All** > **Event Management** > **Rules** > **Event Field Mapping**, and then select **New**.

b. In **Name**, enter your rule name.
The value must match what `getRuleName()` returns. The platform passes this name to the rule script as `fieldMappingRuleName`, and the resolver uses it to find your implementation.

c. Set **Source** to your source name.
This field filters which events the rule applies to.

d. Set **Mapping type** to **Advanced mapping using script**.

e. Set **Active** to true.

f. Set **Order** to a value lower than any existing rule for the same source that you want this rule to take priority over.
Lower values run first. The default rule runs at order 100. To make your rule run first, set a value below 100, for example, 50.

g. In **Filter**, add the condition [Source] [is] [your source name].

h. In the **Script** field, retrieve your implementation through the resolver and call `mapCi()` on it:

```
(function eventFieldMappingScript(eventGr, origEventSysId,
 fieldMappingRuleName) {
    var impl = new
 sn_tsom_em_conns.EventFieldMappingResolver()
        .getByRule('<your source name>', fieldMappingRuleName);
    if (impl) {
        return impl.mapCi(eventGr, origEventSysId,
 fieldMappingRuleName);
    }
    gs.warn('No EventFieldMapping implementation found for rule: ' +
 fieldMappingRuleName);
    return true;
})(eventGr, origEventSysId, fieldMappingRuleName);
```

Replace `<your source name>` with the same value that `getVendor()` returns. If you reuse this script for another source, update it in both places.

i. Submit the rule.

4. Confirm the override.

a. Trigger the connector to collect metrics for the source.
The metric framework creates the metric record and its event automatically.

b. Navigate to **All** > **Event Management** > **Metrics** > **Metric to CI**.
In the **Configuration item** column, verify that new metrics for your source show the CI resolved by your implementation. To trace a binding, open the linked event and verify that the `cmdb_ci` field is set.

c. If the CI is not set, check the system logs for the message "No EventFieldMapping implementation found".
That message means the resolver could not match your `getVendor()` and `getRuleName()` values, or your implementation was created in the wrong application scope. Confirm the names match across the script include and the rule, and that the script include is in the `sn_tsom_em_conns` scope.

Result
New metrics collected from the source are bound to configuration items by your custom implementation.

Related topics

Event field mapping configuration

Create event field mappings

MPN Formulas table

Using Telecommunications Service Operations Management

Leverage Telecommunications Service Operations Management (TSOM) to proactively monitor telecom services, validate data integrity, and reconcile discrepancies across network inventory and discovery sources. TSOM enables operations teams to maintain an accurate telecom-aware CMDB and act on real-time network insights.

Common operational tasks of TSOM

Once TSOM is configured, you can use its capabilities to:

- Run Telecom Discrepancy audits: Identify inconsistencies in network relationships and configurations by running certification audits based on discovered data and reconciliation rules.
- Validate attribute value mismatches in CMDB 360: Use the Attribute Value Discrepancy feature in CMDB 360 to compare attribute values across different discovery sources and detect conflicts or outdated records.

- Use Telecom Discovery Patterns: Apply Telecom Discovery Patterns to accurately discover and populate telecom-specific Configuration Items (CIs) and their relationships.
- Run and verify import jobs from service graph connectors: Manually trigger import schedules (e.g., for the Nokia Altiplano connector) to verify that topology and device data is successfully imported into the CMDB and mapped to the correct CI classes.

These tasks help ensure that your telecom network is continuously synchronized, accurate, and operationally visible within ServiceNow.

Use Telecom Discovery patterns

Use Telecom Discovery patterns to identify and classify network functions (xNFs) from vendor devices such as Cisco, Juniper, routers, or switches. Leverage pattern-based discovery to map telecom resources into the CMDB.

Classify xNFs for Telecom Router Pattern

To access a full list of OIDs that will be classified.

Before you begin

Role required: admin

Classifier name: **Standard Network Router**.

Procedure

1. Navigate to **All** > **Discovery Definition** > **CI Classification** > **SNMP**.
2. From the list, select **Standard Network Router**.
3. Open the tab **SNMP OID Classifications** and see the list of OIDs.

Note:

For more information on how to add additional OIDs to the classifier, see Direct Discovery using Discovery Patterns.

MiB Tables Used on an xNF:

- SystemMIB
- EntityPhysicalMIB
- IfMIB
- IfXMIB
- IpMIB

Classify xNFs for Telecom Cisco 7613 Router Pattern

To access a full list of OIDs that will be classified.

Before you begin

Role required: admin

Classifier name: **Standard Network Router**.

Procedure

1. Navigate to **All** > **Discovery Definition** > **CI Classification** > **SNMP**.
2. From the list, select **Standard Network Router**.
3. Open the tab **SNMP OID Classifications** and see the list of OIDs.

Note:

For more information on how to add additional OIDs to the classifier, see Direct Discovery using Discovery Patterns.

The List of Specific OIDs to call this Pattern:

Vendor	Model	OID	Pattern
Cisco	7613	1.3.6.1.4.1.9.1.528	Telecom Cisco 7613 Router

MiB Tables Used on an xNF:

- SystemMIB
- EntityPhysicalMIB
- IfMIB
- IfXMIB
- IpMIB

Classify xNFs for Telecom Juniper MX SSH Router Pattern

To access a full list of OIDs that will be classified.

Before you begin

Role required: admin

Classifier name: **Standard Network Router**.

Procedure

1. Navigate to **All** > **Discovery Definition** > **CI Classification** > **SNMP**.
2. From the list, select **Standard Network Router**.
3. Open the tab **SNMP OID Classifications** and see the list of OIDs.

Note:

For more information on how to add additional OIDs to the classifier, see Direct Discovery using Discovery Patterns.

The List of Specific OIDs to call this Pattern:

Vendor	Model	OID	Pattern
Juniper	MX80	1.3.6.1.4.1.2636.1.1.1.2.57	Telecom Juniper MX SSH Router
Juniper	MX104	1.3.6.1.4.1.2636.1.1.1.2.97	Telecom Juniper MX SSH Router
Juniper	MX240	1.3.6.1.4.1.2636.1.1.1.2.29	Telecom Juniper MX SSH Router

Juniper	MX480	1.3.6.1.4.1.2636.1.1.1.2.25	Telecom Juniper MX SSH Router

MiB Tables Used on an xNF: SystemMIB.

CLI Commands Used.

- show chassis hardware | no-more | display xml
- show interface media | no-more | display xml

Classify xNFs for Telecom Cisco Switch Pattern

To access a full list of OIDs that will be classified.

Before you begin

Role required: admin

Classifier name: **Standard Network Switch**.

Procedure

1. Navigate to **All** > **Discovery Definition** > **CI Classification** > **SNMP**.
2. From the list, select **Standard Network Switch**.
3. Open the tab **SNMP OID Classifications** and see the list of OIDs.

Note:

For more information on how to add additional OIDs to the classifier, see Direct Discovery using Discovery Patterns.

The List of Specific OIDs to call this Pattern:

Vendor	Model	OID	Pattern
Cisco	Nexus 9000	1.3.6.1.4.1.9.12.3.1.3.1954	Telecom Cisco Switch
Cisco	Nexus 3548	1.3.6.1.4.1.9.12.3.1.3.1666	Telecom Cisco Switch

MiB Tables Used on an xNF:

- SystemMIB
- EntityPhysicalMIB
- IfMIB
- IfXMIB
- IpMIB

Classify xNFs for Telecom Switch Pattern

To access a full list of OIDs that will be classified.

Before you begin

Role required: admin

Classifier name: **Standard Network Switch**.

Procedure

1. Navigate to **All** > **Discovery Definition** > **CI Classification** > **SNMP**.
2. From the list, select **Standard Network Switch**.
3. Open the tab **SNMP OID Classifications** and see the list of OIDs.

Note:

For more information on how to add additional OIDs to the classifier, see Direct Discovery using Discovery Patterns.

MiB Tables Used on an xNF:

- SystemMIB
- EntityPhysicalMIB
- IfMIB
- IfXMIB
- IpMIB

Run SGC imports

Run and verify a Nokia Altiplano import

Manually execute a configured import schedule for the Nokia Altiplano Service Graph Connector. You can also execute the import to validate the connector setup, run ad-hoc imports, or test newly configured connection aliases. This task helps ensure that data from Altiplano is successfully imported or updated in the CMDB.

Before you begin

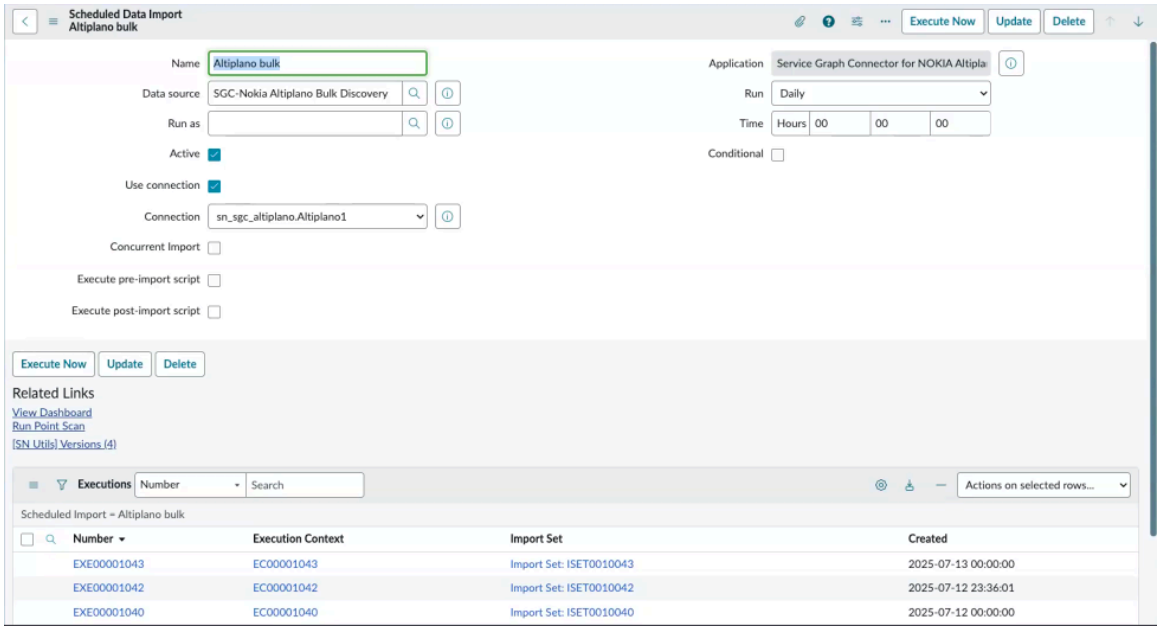
Role required: TSOM Visibility admin

About this task

After configuring the Nokia Altiplano Service Graph Connector and setting up one or more import schedules, you can execute an import job. This task helps you verify that the integration is working as expected and that CIs are successfully imported or updated in the CMDB.
You can either enable the import schedules to run automatically based on their defined frequency, or trigger them manually for immediate execution and validation.

Note: If you have configured multiple Altiplano connection aliases (for different environments or filters), confirm that the import schedule is set up correctly for each alias.

The following screenshot helps you understand the scheduled data import process and displays the executions in the form of import sets.



Procedure

1. Navigate to **All** > **Nokia Altiplano** > **Import Schedules**.
2. Select the import schedule that you want to run.
   The list of scheduled data imports appear.
3. Select the scheduled data import.
4. Do one of the following:
   - If the schedule is **Active**, the job runs automatically at the defined interval. So wait until it's executed.
   - Select **Execute Now** to run the import immediately.
5. To monitor the execution and verify results:
   a. Scroll down to the **Executions** list or the table linked to the import schedule.
   b. Open the most recent **Import Set** record created by the execution.
6. Review the import set log to verify the numbers of rows read, number of rows inserted or updated in the CMDB, or transformation success status.
7. **Optional:** Navigate to **All** > **CMDB** > **CI Classes** or your custom telecom tables (such as TNI base item) to confirm the CIs were created or updated as expected.

Result

The import schedule runs immediately, and you can verify the import set execution and the corresponding CI updates in the CMDB. If configured correctly, the connector brings in the network inventory data from Nokia Altiplano into your ServiceNow instance.

Example: After executing the import schedule:

- The log shows: "12 rows read, 0 inserted/updated." This indicates that CIs exist.
- If you delete existing CIs and rerun the import, the log might show multiple inserts and updates, validating end-to-end functionality.

Note:

- If **Concurrent Import** option is enabled in the import schedule, you'll see records in the **Concurrent Import Set** table instead of the standard Import Set table.
- Each concurrent job creates its own import set and log entry. The structure of the execution records remains the same.

Related topics

Configure Nokia Altiplano service graph connector

Telecom Discovery via Nokia Altiplano

Run and verify an import schedule for Fortinet SGC

Manually run a configured import schedule for the Fortinet Service Graph Connector (SGC) to verify that data from Fortinet was successfully imported or updated in the Configuration Management Database (CMDB). You can also perform the import to validate the connector setup, run one-off imports, or test newly configured connection aliases.

Before you begin

Role required: TSOM Visibility admin

If you have configured multiple Fortinet connection aliases for different environments or filters, confirm that the import schedule is set up correctly for each alias.

About this task

Note:

- If Concurrent Import option is enabled in the import schedule, records are imported into the Concurrent Import Set table instead of the standard Import Set table.
- Each concurrent job creates its own import set and log entry. The structure of the execution records remains the same.

Procedure

1. Navigate to **All** > **Fortinet** > **Import Schedules**.
2. Select the import schedule that you want to run.
3. From the list of scheduled data imports, select the scheduled data import.
4. If the import needs to be run manually rather than having run automatically on a specified schedule, select **Execute now**.
5. In the Executions related list, verify the import results by selecting the most recent Import Set record created by the execution of the import.
6. Review the import set log to verify the numbers of rows read, number of rows inserted or updated, or transformation success status, in the CMDB, which indicate that the CIs exist. If you delete existing CIs and rerun the import, the log might show multiple inserts and updates, validating end-to-end functionality.

Result

If configured correctly, the connector brings in the network inventory data from Fortinet into your ServiceNow AI Platform® instance.

Run and verify an import schedule for Cisco Meraki SGC

Manually run a configured import schedule for the Cisco Meraki Service Graph Connector (SGC) to verify that data from Cisco Meraki was successfully imported or updated in the Configuration

Management Database (CMDB). You can also perform the import to validate the connector setup, run one-off imports, or test newly configured connection aliases.

Before you begin

Role required: TSOM Visibility admin

If you have configured multiple Cisco Meraki connection aliases for different environments or filters, confirm that the import schedule is set up correctly for each alias.

About this task

Note:

- If Concurrent Import option is enabled in the import schedule, records are imported into the Concurrent Import Set table instead of the standard Import Set table.
- Each concurrent job creates its own import set and log entry. The structure of the execution records remains the same.

Procedure

1. Navigate to **All** > **Meraki** > **Import Schedules**.
2. Select the import schedule that you want to run.
3. From the list of scheduled data imports, select the scheduled data import.
4. If the import needs to be run manually rather than having run automatically on a specified schedule, select **Execute now**.
5. In the Executions related list, verify the import results by selecting the most recent Import Set record created by the execution of the import.
6. Review the import set log to verify the numbers of rows read, number of rows inserted or updated, or transformation success status, in the CMDB, which indicate that the CIs exist. If you delete existing CIs and rerun the import, the log might show multiple inserts and updates, validating end-to-end functionality.

Result

If configured correctly, the connector brings in the network inventory data from Cisco Meraki into your ServiceNow AI Platform® instance.

Validating JSON payloads using TSOM Schema Validator

Use the TsomSchemaValidator utility class to validate JSON payloads against TSOM schemas before importing data. This helps identify errors early, reduce ETL failures, and confirm data quality.

Use this validator to check whether your JSON payloads conform to the expected schema for telco objects like Devices, Logical Connections, and Topologies before creating import sets. This pre-validation step helps to prevent schema mismatch errors and improves debugging.

Supported schema types

The validator supports multiple schema types for different telco data structures:

- Logical Composites - Representing groupings of components: Equipment, PDUs, Fan Shelves
- Devices - Equipment and their contained components
- Logical Connections - Connections between network interfaces
- Port Relations - Relationships between network interfaces: physical, logical, lags

- Logical Connection Relations - Relationships between logical connections
- Topologies - Network topology

Class structure

```
let TsomSchemaValidator = Class.create();
TsomSchemaValidator.prototype = {
initialize: function() {
this.schemas = new TsomGenericSchema();
},
isValidJson: function(payload) {
// Validation logic that determines if the JSON structure is
 valid
// Returns boolean (true/false)
},
checkJsonValidation: function(payload) {
// Validation logic that determines if the JSON structure is
 valid
// Returns a JSON object containing errors (if exist)
},
type: 'TsomSchemaValidator'
};
```

Steps

1. Instantiate the Schema ValidatorjavascriptCopyEdit

```
var TsomSchemaValidator = new
 sn_tsom_core.TsomSchemaValidator();
```

2. Run a Boolean Validation Check

```
if (!TsomSchemaValidator.isValidJson(target_json)) {
    gs.error('Invalid JSON: ' + JSON.stringify(target_json));
    return;
}
```

3. Run a Detailed Validation Check

```
let result =
 TsomSchemaValidator.checkJsonValidation(target_json);
if (!result.valid) {
    gs.error('Invalid JSON: ' + JSON.stringify(result, null,
 2));
    return;
}
```

Example output

```
Example Output
{
  "schemaName": "devices",
  "errors": [
    {
      "message": "Missing required property: model_name",
      "params": { "key": "model_name" },
```

```
      "code": 302,
      "dataPath": "/devices/0/ports/0",
      "schemaPath":
 "/properties/devices/items/properties/ports/items/required/4"
    }
  ],
  "valid": false
}
```

Related topics

Configure the Telecom Discovery Builder ETL

Telecom Discovery Builder framework

Run Telecom Discrepancy audit

The Telecom Discrepancy Audit validates the integrity of configuration items (CIs) and relationships across your telecom inventory using the CMDB Compliance framework.

Before you begin

Role required: admin

About this task

The Telecom Discrepancy Audit is part of the CMDB Compliance framework and supports phased auditing to detect and address discrepancies in CI relationships and attributes discovered through ServiceNow Discovery or Service Graph Connectors (for example, Nokia Altiplano).

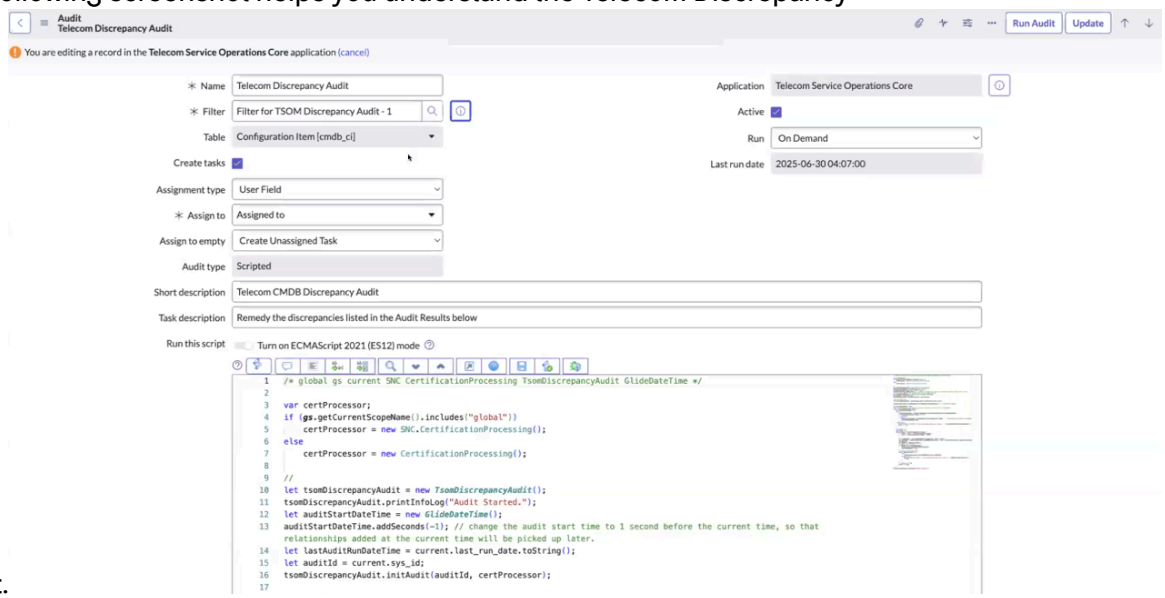
You can run the audit manually or set it to run at regular intervals. During manual execution, you can select from existing filters to limit the audit scope to specific CIs—helpful for testing or targeted reconciliation.

The Telecom Discrepancy Audit works in two phases:

1. Initial compliance run: Validates CI relationships and selected CI tables using the following default criteria:
   - Supported CI classes: Slot, Subslot, Card, Interface, Network Gear (includes all extension tables).
   - Source: CI was discovered via Discovery (For example `discovery_source = SG-TSOM-Altiplano`).
   - Relationship type: `Contains::Contained By` (customizable via `sn_tsom_core.audit.relationship_types` property).
2. Subsequent compliance runs: In addition to initial checks, evaluates whether the Updated timestamp of the relationship or related CIs is more recent than the Last run date in the audit record.

> **Note:** Each failed audit creates a Follow-On Task for manual or automated remediation, confirming telecom CMDB data stays aligned with the network state.

The following screenshot helps you understand the Telecom Discrepancy Audit.



Procedure

1. Navigate to **All** > **Compliance** > **Audits**
2. Open the Telecom Discrepancy Audit (cert_audit table) record.
3. **Optional:** Select a filter.
   - If multiple audit filters have been defined during configuration, you can choose one before running the audit.
   - Filters enable you to restrict the audit scope based on criteria such as discovery source, CI class, or specific CI attributes.
   - This step is especially useful when troubleshooting a subset of records or validating specific discovery results.
4. Do either of the following to run the audit.
   - Click **Run Audit** to trigger a manual execution.
   - Configure the audit to run on a scheduled basis by setting up a recurring job.

   The audit uses CMDB Compliance to validate relationship and CI integrity across telecom inventory.

Result

Review audit results

- The audit identifies compliance failures such as missing relationships or misaligned attributes.
- For each failed audit record, a Follow-On Task is automatically generated.
- Tasks include recommended or automated remediation actions such as updating or decommissioning CIs.

Example use case: The configured filters can be used for different discovery sources. For example, the Nokia Altiplano. When running the audit manually, you can select the appropriate filter to validate only those CIs discovered by that specific source—confirming targeted and efficient auditing.

Related topics

Configure filter for audit

Example for Telecom Discrepancy Audit and Remediation

Example for Telecom Discrepancy Audit and Remediation

The following example illustrates how the audit works in a scenario.

Before you begin

Role required: admin

About this task

A card (Card04) was initially discovered in Slot04. Later, Card04 was replaced by Card05 in the physical network, but the CMDB still shows Card04 in Slot04. When a new discovery run executes, Card05 is discovered and added to the same slot, creating a data conflict in CMDB.

Audit Behavior:

- The Telecom Discrepancy Audit detects this inconsistency and creates a failed audit record (e.g., AUDR0001283).
- A Follow-On Task is created automatically (e.g., TASK0020215) with a detailed description of the discrepancy.

Task Description Example (Incorrect Number of Relationships):

Card04 was last discovered more than 2.5 days ago.

Relationships between the following CIs

CI	Model
Slot04	DEMO 20532Tree
Card04	Nokia 7360 FANT-F CARD MODULE
Card05	<model not identified>

Procedure

1. Navigate **All** > **Compliance** > **Audits** > **Telecom Discrepancy Audit** > **Run Audits**.
   A Follow-on Task is automatically created for each failed audit record (for example, TASK0020215).
2. Select **TASK0020215** and review the discrepancy description.

   Note: This is an example of the TASK0020215 description created for the "Incorrect number of relationships" scenario. Other scenarios and environments might have different descriptions.

   The Follow-On Task contains a detailed description of the discrepancy. As you can see in the description, the Card04 CI is in discrepancy.
3. To remediate, perform the following steps:

   a. Navigate **All** > **System Definition** > **UI Actions**.

   b. Select **Remediate**.

The Remediate button is a UI action that triggers the `TSOM CI Decommission` subflow. This subflow:

- Retires the outdated Card04 CI.
- Removes the incorrect Slot04 → Card04 relationship.
- Synchronizes CMDB records with the network state.

Result

After remediation:

- The Follow-On Task (`TASK0020215`) updates with work notes summarizing the resolution steps.
- The Card04 CI is marked as retired, and incorrect relationships are removed.
- CMDB is now aligned with the latest discovered state from the network.

Note: Customize Remediation - The Telecom Discrepancy Audit ships with example remediation subflows. You can create and attach custom subflows using Flow Designer to suit your operational requirements.

Related topics

Telecom Discrepancy Identification and Reconciliation

Discrepancy identification – types of discrepancies

Telecom discrepancy identification and reconciliation

Activate Telecom Discrepancy Identification and Reconciliation

Defining UI actions

Generate reports for attribute value discrepancies

Use CMDB 360 to generate reports that highlight discrepancies in attribute values between different discovery sources or between a discovery source and the CMDB baseline.

Before you begin

Role required: admin

About this task

Attribute value discrepancy reports help identify conflicting data from multiple sources updating the same configuration item (CI). This enables better data quality and integrity across your CMDB.

Procedure

1. Navigate to **All** > **CMDB Workspace** > **CMDB 360**.
2. Select **Create Query**.
3. In the query type window, select **Compare attribute values**.
4. Define the query parameters:
   a. Select **CI Class** you want to analyze (for example, cmdb_ci_computer).
   b. Apply filters to narrow the scope (for example, Discovery source contains "TSOM").
   c. Select attributes to compare values across different discovery sources.
   d. Choose the sources to compare (for example, Discovery, SCCM).

e. **Optional:** Select **Compare to CMDB** to compare discovery source data to the current CMDB baseline.

f. Click **Save**, provide a name for the query, and then click **Run** to generate the results.

5. **Optional:** Click **Schedule** to set up the query to run at defined intervals.

Result

The report displays a comparison of attribute values across sources:

- Source 1: Current CMDB record.
- Source 1 Value: Value updated by the most recent discovery source.
- Source 2: Previous discovery source.
- Source 2 Value: Value provided by the earlier discovery source.

Each discrepancy is listed as a separate record when a CI has been updated by multiple sources.

Related topics

Configure attribute value discrepancy in CMDB 360

Control CI attribute updates using Reconciliation rules

Telecommunications Service Operations Management reference

Several types of components are installed with Telecommunications Service Operations Management applications and plugins.

Fortinet installed integrations

Predefined system integrations use Fortinet REST APIs to pull metric data into your ServiceNow instance to monitor your Fortinet devices.

Fortinet installed integrations

Fortinet integration	Description
Performance status	Monitors resource utilization and system health for FortiGate devices, including CPU, memory, and session data.
SD-WAN SLA log	Collects SD-WAN link performance data, including latency, jitter, and packet loss against configured SLA thresholds.

Cisco Meraki installed integrations

Predefined system integrations use Meraki REST APIs to pull metric data into your ServiceNow instance to monitor your Cisco Meraki devices.

Cisco Meraki installed integrations

Cisco Meraki integration	Description
Appliance performance	Collects performance metrics from MX and Z series appliances, including device utilization, uplink bandwidth usage, and uplink loss and latency.
Device Performance	Collects performance metrics from devices, including utilization, throughput, and availability data.
Uplink Statuses	Monitors the connectivity status of WAN uplinks on MX and Z series devices.
Configuration Changes	Tracks configuration changes made to devices and networks in the dashboard.
Network Shaping Uplink Bandwidth	Collects configured bandwidth limits for WAN uplinks used in traffic shaping.
Switch Port Statuses by Switch	Monitors port status and connectivity for MS switches.
Uplink Usage by Network	Collects uplink bandwidth usage data aggregated by network for MX and Z series devices.
Uplink Loss and Latency	Monitors packet loss and latency for WAN uplinks on MX appliances.
VPN Stats	Monitors site-to-site VPN tunnel status for MX appliances.

System components installed with Telecommunications API notifications

Administrators can assign user roles to grant access to the API notification database tables. The following standard roles for the Topic [sn_api_notif_mgmt_topic] and Topic Subscription [sn_api_notif_mgmt_subscription] tables are included in the ServiceNow system.

Telecommunications API notification roles

Role	Description
sn_api_notif_ mgmt.topic_subscription_viewer	Role that enables with read access to the Topic and Topic Subscription tables.
sn_api_notif_ mgmt.topic_creator	Role that enables with create, read, and edit access to the Topic table.
sn_api_notif_mgmt.subscription_creator	Role that enables with create and read access to the Topic Subscription table.
sn_api_notif_mgmt.subscription_admin	Role that enables with the following permissions:

Telecommunications API notification roles (continued)

Role	Description
	• Create and read access to the Topic and Topic Subscription tables. • Change the status of registration to deregister a topic subscription.

Related topics

External event management via Telecommunications API notifications

Configuring Telecommunications API notifications

System components installed with Nokia Altiplano

System properties control how the connector operates, including discovery options and performance settings.

Properties installed with Nokia Altiplano

Property	Description
sn_sgc_altiplano.enable_onu_discovery	Enable or disable discovery of ONU devices and logical connections between OLT and ONU. • Default value: True • Location: **All**>**Service Graph Connectors**>**Nokia Altiplano**>**Properties** or **System Properties** [sys_properties] table filtering by the name '*altiplano'
sn_sgc_altiplano.devices_list_batch_size	Controls the batch size for Altiplano REST API calls. • Default value: 1000 • Location: **All**>**Service**

Properties installed with Nokia Altiplano (continued)

Property	Description
	Graph Connectors>**Nokia Altiplano**>**Properties** or **System Properties** [sys_properties] table filtering by the name '*altiplano'
sn_sgc_altiplano.parallel_number_of_data_source_jobs	Number of parallel jobs for collecting Altiplano data (requires "Enable parallel loading" setting). • Default value: 2 • Location: **All**>**Service Graph Connectors**>**Nokia Altiplano**>**Properties** or **System Properties** [sys_properties] table filtering by the name '*altiplano'
sn_sgc_altiplano.onu_ci_class	Defines whether ONUs are stored as ONU or ONT CI class. • Default value: ONU • Location: **All**>**Service Graph Connectors**>**Nokia Altiplano**>**Properties** or **System Properties** [sys_properties] table filtering by the name '*altiplano'

Related topics

Telecom Discovery via Nokia Altiplano

Configure Nokia Altiplano service graph connector

Telecom discrepancy identification and reconciliation

The system properties are part of the TSOM Visibility plugin (sn_tsom_core) and control the Telecom Discrepancy Identification & Reconciliation log (TSOM CMDB Audit). The TSOM Visibility plugin serves as an enabler for the TSOM Visibility applications, containing logic that is shared across the Telecom Discovery and Telecom Discrepancy Identification & Reconciliation solution.

Telecom Discrepancy Identification & Reconciliation System Properties (Impacts CMDB Audit)

Property Name	Recommended Default Value	Description
sn_tsom_core.audit.log.level	Telecom Discrepancy Audits' Logging.	debug,info,warn Default value: info
sn_tsom_core.audit.relationship_types	Displayed Names of Relationship Types that will be handled by Telecom Discrepancy Audit.	Comma separate Default value: Co
sn_tsom_core.audit.discovered_date.diff.threshold.in.days	Most recent discovered date threshold in days. Used for Telecom Discrepancy Audit.	String Default value: 2.5
sn_tsom_core.audit.suppress_CI_Model_missing_discrepancy_task_creation	Suppress creation of CI Model missing discrepancy task for the tables.Used for Telecom Discrepancy Audit.	cmdb_ci_ni_telco cmdb_ci_contain
sn_tsom_core.audit.equipment_tables	Equipment tables that will be handled by	Comma separate Default value: cm

Telecom Discrepancy Identification & Reconciliation System Properties (Impacts CMDB Audit) (continued)

Property Name	Recommended Default Value	Description
	Telecom Discrepancy Audit.	
sn_tsom_core.audit.interface_card_tables	Interface Cards tables that will be handled by Telecom Discrepancy Audit.	Comma separate Default value: cm
sn_tsom_core.audit.interface_tables	Network Interface tables that will be handled by Telecom Discrepancy Audit.	Comma separate Default value: cm
sn_tsom_core.audit.slot_tables	Slot tables names that will be handled by Telecom Discrepancy Audit.	Comma separate Default value: cm
sn_tsom_core.audit.subslot_tables	Subslot tables names that will be handled by Telecom Discrepancy Audit.	Comma separate Default value is c

Telecom Discrepancy Identification & Reconciliation Audits

Audit Name	Description
Telecom Discrepancy Audit	Audits records in the `cmdb_rel_ci` table. For each relationship record dis or NSP, it validates the parent and child CIs based on: - Most recent discover relationship (if TNI Core is installed). **Note:** All discrepancy audits use a filtering mechanism that applies to e filter condition is: `discovery_source CONTAINS TSOM`.
Telecom Logical Connections Discrepancy Audit	Audits records in the `cmdb_ci_ni_logical_path` table. For each logica Nokia Altiplano or NSP, it checks: - Empty endpoint (Port A or Port Z) - Each e only one unique logical connection.

Telecom Discrepancy Identification & Reconciliation Audits (continued)

Audit Name	Description
Telecom Network Topology Discrepancy Audit	Audits records in the `cmdb_ci_network_topology` table. For each netw by Nokia Altiplano or NSP, it verifies: - At least one Contains:Contained By rel least one Members:Member Of relationship with a logical connection.

Related topics

- Telecom Discrepancy Identification and Reconciliation
- Activate Telecom Discrepancy Identification and Reconciliation
- Run Telecom Discrepancy audit

Fortinet Service Graph Connector API Endpoints

The Service Graph Connector for Fortinet integrates Fortinet Dashboard API data into ServiceNow AI PlatformConfiguration Management Database (CMDB). This document details the API endpoints used and how data flows through the system.

Note: The `meta fields` values in the ADOM and device requests are configurable. Define the meta field names that match your Fortinet deployment. If you don't configure any meta fields, the connector omits the `meta fields` parameter from the request, and Fortinet returns standard data. The values shown in the following examples are sample meta fields and are required for discovery.

API reference examples

Description	API response
Authenticate JSON-RPC session	`{ "method": "exec", "params": [ { "url": "/sys/login/user", "data": { "user": "user", "passwd": "abc1123" } } ], "verbose": 1 }`
Logout JSON-RPC session	`{ "method": "exec", "params": [ { "url": "/sys/logout" } ], "verbose": 1, "session": "TxMw/zbjw+tVZ/JL3bhmYg0vTUpVguYHIQesJXpye4j2vvsqtZaSgKWa+0iLqug+3/074jq8QqmI/KOx4GHkwQ==" }`
GET ADOMs	`{ "method": "get", "params": [ { "url": "/dvmdb/adom", "meta fields": [ "VF SDWAN Service Provided", "VF OpCo Name", "VF Customer ID", "SERVICE_ID", "SDN_MS" ] } ], "verbose": 1, "session": "TxMw/zbjw+tVZ/JL3bhmYg0vTUpVguYHIQesJXpye4j2vvsqtZaSgKWa+0iLqug+3/074jq8QqmI/KOx4GHkwQ==" }`

API reference examples (continued)

Description	API response
GET ADOM devices	{ "method": "get", "params": [{ "url": "/dvmdb/adom/{ADOM_NAME}/device", "meta fields": ["Company/Organization", "VF 3C Ref", "Address", "Contact Email", "Contact Phone Number", "VF Order Ref"] }], "verbose": 1, "session": "TxMw/zbjw+tVZ/JL3bhmYg0vTUpVguYHIQesJXpye4j2vvsqtZaSgKWa+0iLqug+3/074jq8Qqml/KOx4GHkwQ==" }
GET interfaces by device name	{ "method": "get", "params": [{ "url": "/dbcache/system/interface", "option": ["scope member"], "scope member": [{ "name": "{DEVICE_NAME}", "vdom": "global" }], "fields": ["name", "alias", "comments", "speed", "mediatype", "ip", "mode", "type", "mtu", "interface", "vlanid", "member", "vdom", "role", "allowaccess", "zone", "mac", "status"], "current_adom": "{ADOM_NAME}" }], "verbose": 1, "session": "TxMw/zbjw+tVZ/JL3bhmYg0vTUpVguYHIQesJXpye4j2vvsqtZaSgKWa+0iLqug+3/074jq8Qqml/KOx4GHkwQ==" }

Cisco Meraki Service Graph Connector API Endpoints

The Service Graph Connector for Meraki integrates Cisco Meraki Dashboard API data into ServiceNow AI Platform® Configuration Management Database (CMDB). This document details the API endpoints used and how data flows through the system.

Organizations endpoint

Description	API response

Organizations API response
 	
URL: `/organizations`
	{
	"id": "123456",
	"name": "My Organization",
	"management": {

Organizations endpoint (continued)

Description	API response
	"details": { "MSP ID": "...", "customer number": "...", "IP restriction mode for API": "..." } }

Networks Endpoint

Description	Endpoint
Networks API response URL: `/organizations/{orgId}/networks`	{ "id": "N_123456", "organizationId": "123456", "name": "My Network", "notes": "Network description" }

Devices Endpoint

Description	Endpoint
Devices API response URL: `/organizations/{orgId}/devices`	{

Devices Endpoint (continued)

Description	Endpoint
	"serial": "Q2XX-XXXX-XXXX", "name": "My Device", "networkId": "N_123456", "model": "MX68", "mac": "00:11:22:33:44:55", "productType": "appliance", "firmware": "mx-18.1", "lat": 37.7749, "lng": -122.4194, "address": "123 Main St", "notes": "Device description" }

Device statuses endpoint

Description	Endpoint
Devices statuses API response `URL: /organizations/{orgId}/devices/statuses`	{ "serial": "Q2XX-XXXX-XXXX", "status": "online", "publicIp": "203.0.113.1",

Device statuses endpoint (continued)

Description	Endpoint
	"wan1Ip": "192.168.1.1", "wan2Ip": "192.168.2.1", "wan1Gateway": "192.168.1.254", "wan1IpType": "dhcp", "wan1PrimaryDns": "8.8.8.8", "wan1SecondaryDns": "8.8.4.4" }

Uplink statuses endpoint

Description	Endpoint

Uplink statuses endpoint
 	
`URL: /organizations/{orgId}/uplinks/statuses`
	{
	"serial": "Q2XX-XXXX-XXXX",
	"networkId": "N_123456",
	"model": "MX68",
	"highAvailability": {
	"enabled": true,
	"role": "primary"
	},
	"uplinks": [

Uplink statuses endpoint (continued)

Description	Endpoint
	{ "interface": "wan1", "status": "active", "ip": "192.168.1.1", "gateway": "192.168.1.254", "publicIp": "203.0.113.1", "primaryDns": "8.8.8.8", "secondaryDns": "8.8.4.4", "ipAssignedBy": "dhcp" }, { "interface": "cellular", "status": "ready", "apn": "broadband", "iccid": "...", "imsi": "...", "msisdn": "..."

Uplink statuses endpoint (continued)

Description	Endpoint
	}] }

Switch ports endpoint

Description	Endpoint
Switch ports endpoint `URL: /organizations/{orgId}/switch/ports/statuses/bySwitch`	{ "items": [{ "serial": "Q2XX-XXXX-XXXX", "network": { "id": "N_123456" }, "ports": [{ "portId": "1", "status": "connected" }, { "portId": "2",

Switch ports endpoint (continued)

Description	Endpoint
	"status": "disconnected" }] }] }

Device inventory endpoint

Description	Endpoint

Device inventory endpoint
 	
`URL: /organizations/{orgId}/inventory/devices`
	{
	"serial": "Q2XX-XXXX-XXXX",
	"orderNumber": "ORD-12345",
	"licenseExpirationDate": "2025-12-31"
	}

MPN Formulas table

Reference for the MPN Formulas [sn_tsom_em_conns_kpi_definitions] table and the automatic processing that occurs when a record is inserted or updated.

MPN Formulas table columns

MPN Formulas table columns

Column	Description
Release	Release version
Vendor	Vendor name
Technology	Radio technology type (for example, NR, LTE)

MPN Formulas table columns (continued)

Column	Description
KPI Formula	Raw formula string
Index	Elasticsearch index pattern to query (for example, `radio-pm*`)
Labels Up Down	Indicates the direction of the KPI metric. The available values are up and down.
Labels Vendor KPI ID	Unique KPI identifier assigned by the vendor
Labels KPI Standard Name	Standardized name for the KPI
Labels KPI Node	Node-level label associated with the KPI. The available values are RAN, Core, or Latency Collector.
Unit	Unit of measurement for the KPI value, for example, %, Mbps
Source Time Interval	Data collection time interval from the source system
Aggregation Type	How the KPI value is aggregated, for example, AVG, SUM
CI Label	Path in the Elasticsearch document used to extract the CI identifier
CI Class	ServiceNow CI class the KPI is mapped to
Component Aggregation	Aggregation method applied at the component level. The available values are flat, hierarchy, or flat+hierarchy.
MPC Description	Full description of the KPI from the MPC specification
MPC UX Name	Display name for the KPI as shown in the MPC UI
MPC KPI Group	Top-level KPI grouping as defined in the MPC
MPC KPI Sub Group	Sub-grouping of the KPI within the MPC group
Formatted KPI Formula	Auto-populated. Do not enter a value manually.

Related topics

MPN Formula Engine processing