Cisco WAE 7.5.0 User Guide

First Published: 2021-10-29

Last Modified: 2023-10-23

CONTENTS

Overview

This section contains the following topics:

Cisco WAE Overview

The Cisco WAN Automation Engine (WAE) platform is an open, programmable framework that interconnects software modules, communicates with the network, and provides APIs to interface with external applications.

Cisco WAE provides the tools to create and maintain a model of the current network through the continual monitoring and analysis of the network and the traffic demands that is placed on it. At a given time, this network model contains all relevant information about a network, including topology, configuration, and traffic information. You can use this information as a basis for analyzing the impact on the network due to changes in traffic demands, paths, node and link failures, network optimizations, or other changes.

Some of the important features of Cisco WAE platform are:

- Traffic engineering and network optimization—Compute TE LSP configurations to improve the network performance, or perform local or global optimization.

- Demand engineering—Examine the impact on network traffic flow of adding, removing, or modifying traffic demands on the network.

- Topology and predictive analysis—Observe the impact to network performance of changes in the network topology, which is driven either by design or by network failures.

- TE tunnel programming—Examine the impact of modifying tunnel parameters, such as the tunnel path and reserved bandwidth.

- Class of service (CoS)-aware bandwidth on demand—Examine existing network traffic and demands, and admit a set of service-class-specific demands between routers.

Cisco WAE Architecture

At its core, Cisco WAE defines an *abstract network model*, which can be built from an actual network by stitching together *network interface modules (NIMOs)*.

The Cisco WAE network model is defined in SQLite and is *extensible* via standard SQLite mechanisms. WAE itself is implemented on top of a YANG run-time system that automatically generates APIs (NETCONF, RESTConf, CLI) from the YANG models.
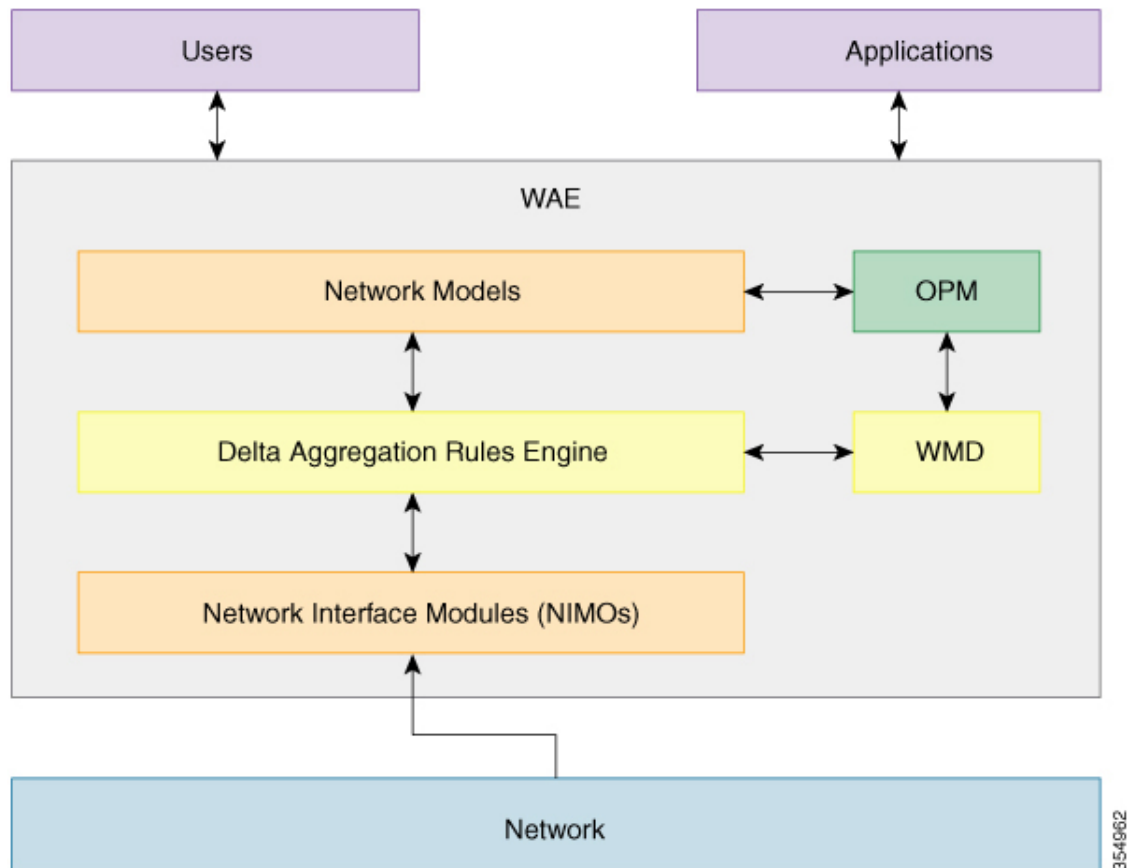
Network Interface Modules

A *network interface module (NIMO)* is a WAE package that populates parts of the abstract network model, querying the network to do so. Most NIMOs operate as follows:

1. They read a *source network model* (or simply, a *source model*).

2. They augment the source model with information obtained from the actual network.

3. They produce a *destination network model* (or simply, a *destination model*) with the resulting model.

WAE includes several different NIMOs, such as:

- Topology NIMO—Populates a basic network model with topology information (nodes, interfaces, circuits) based on the discovered IGP database augmented by SNMP queries. The topology NIMO does not have a source model.

- LSP configuration NIMO—Augments a source model with LSP information, producing a destination model with the extra information.

- Traffic poller NIMO—Augments a source model with traffic statistics polled from the network, producing a new destination model with extra information.

- Layout NIMO—Adds layout properties to a source model to improve visualization. It produces a new destination model with the extra layout information. The NIMO records changes to the layout properties, so when the source model changes and the destination model is updated, the layout properties in the destination model are updated accordingly.

For a comprehensive list of all the NIMOs supported by WAE, see Network Interface Modules (NIMOs), on page 55

Network Models

A *model building chain* is an arrangement of NIMOs organized in such a way as to produce a network model with the desired information.

Delta Aggregation Rules Engine

The DARE aggregator is a WAE component that brings together various NIMOs, selecting model information from each of them, and consolidating the information into a single model. DARE first consolidates any configured topology NIMOs, creates a model, then runs other NIMOs against that model. For example, DARE consolidates an LSP configuration NIMO, L3 topology NIMO, L1 topology NIMO into a single model. It is then followed by traffic collection, inventory collection, layout, netflow and demands.

The following diagram shows a chain tied together by the DARE aggregator:
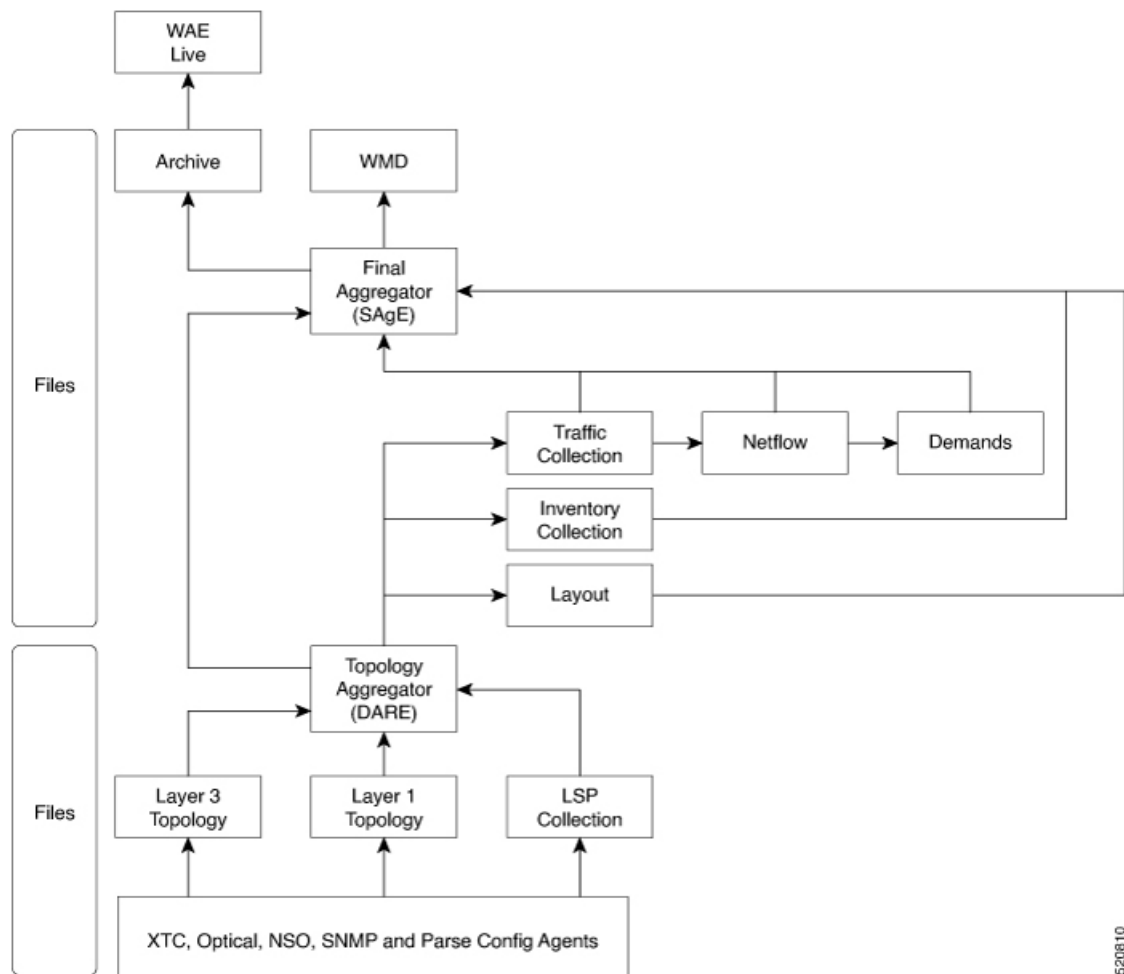
Note   Since DARE works and is based off of changes, it should be configured before changes are made to NIMO models.

For information on how to configure the aggregator to use DARE, see NIMO Collection Consolidation, on page 62.

Simple Aggregation Engine

Simple Aggregation Engine (SAgE) is a WAE component which consolidates all the network information such as traffic, inventory, layout, netflow, demands and aggregates these changes along with the topology changes from DARE network into the final network. The network information from all the NIMOs is written into plan files. The network changes can be archived from SAgE.

SAgE aggregator enables to run traffic collection, inventory collection, layout, etc in parallel.

For information on how to configure the SAgE aggregator, see Run Traffic Collection or a Custom Script Using the Network Model Composer, on page 21

Cisco WAE Modeling Daemon (WMD)

WMD receives changes from SAgE, incorporating scheduled NIMO runs. All updates are consolidated into a near real-time primary model of the network. Cisco WAE applications (described in the next section) are able to connect to WMD and gain access to a copy of this near real-time model in order to leverage Cisco WAE OPM API functionality. WMD configuration is optional and is only required when using and Bandwidth applications.

For information on how to configure WMD, see Configure the WAE Modeling Daemon (WMD), on page 93.

Cisco WAE Applications

Cisco WAE provides a flexible and powerful application development infrastructure. A simple Cisco WAE application consists of:

- The application interface, defined in a YANG model. This interface usually includes RPCs and data models. The YANG models can, if necessary, extend the Cisco WAE network model, adding new data types.

- The application logic, implemented using the *Optimization and prediction modules (OPMs).*

  OPM APIs provide a powerful Python API to manipulate network models. It lets you operate on the network without having to worry about device-specific properties. Even if the underlying routers are replaced by routers from a different vendor, the API calls remain exactly the same.
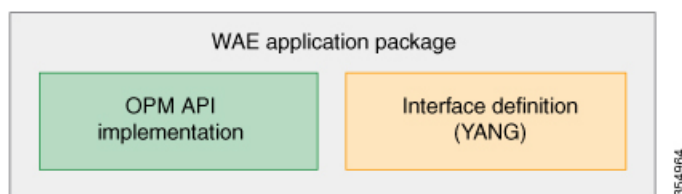
Because Cisco WAE automatically generates APIs from YANG definitions, a Cisco WAE application has its APIs automatically exposed. A Cisco WAE application is, in a sense, a seamless extension of Cisco WAE functionality.

Bandwidth on Demand Application

The Bandwidth on Demand (BWoD) application utilizes the near real-time model of the network offered by WMD to compute and maintain paths for SR policies with bandwidth constraints delegated to WAE from XTC. In order to compute the shortest path available for a SR policy with a bandwidth constraint and ensure that path will be free of congestion, a Path Computation Element (PCE) must be aware of traffic loading on the network. The WAE BWoD application extends the existing topology-aware PCE capabilities of XTC by allowing delegation of bandwidth-aware path computation of SR policies to be sub-delegated to WAE through a new XTC REST API. Users may fine-tune the behavior of the BWoD application, affecting the path it computes, through selection of application options including network utilization threshold (definition of congestion) and path optimization criteria preferences.

For information on how to configure the BWoD application, see Bandwidth on Demand Configuration Workflow, on page 141.

Bandwidth Optimization Application

The Bandwidth Optimization application is an approach to managing network traffic that focuses on deploying a small number of LSPs to achieve a specific outcome in the network. Examples of this type of tactical traffic engineering are deploying LSPs to shift traffic away from a congested link, establishing a low-latency LSP for priority voice or video traffic, or deploying LSPs to avoid certain nodes or links. WAE provides the Bandwidth Optimization application to react and manage traffic as the state of the network changes.

For information on how to configure the Bandwidth Optimization application, see Bandwidth Optimization Application Workflow, on page 146.

Cisco WAE Interfaces

Cisco WAE has three interfaces that you can use to configure your network model:

Cisco WAE UI

The Cisco WAE UI provides an easy-to-use interface that hides the complexity of creating a model building chain for a network. The Cisco WAE UI combines the configuration of multiple data collections under one network and can produce a single plan file that contains the consolidated data. However, there are certain operations that cannot be performed with the Cisco WAE UI. Any configurations done using the Expert Mode or Cisco WAE CLI may not appear in the Cisco WAE UI configuration screens. See Network Model Configuration—Cisco WAE UI, on page 9.

Expert Mode

The Expert Mode is a YANG model browser with additional device and service functionality that might not be available in the WAE UI. Users might prefer to use the Expert Mode over the Cisco WAE CLI because all options for each operation are visible in the Expert Mode. See Network Model Configuration—Expert Mode, on page 25.

Cisco WAE CLI

The Cisco WAE CLI is the interface in which the user responds to a visual prompt by typing a command; a system response is returned. It is the bare-bones interface for all Cisco WAE configurations. Operations available in the Expert Mode are also available in the Cisco WAE CLI. See Network Model Configuration—Cisco WAE CLI, on page 37.

Network Model Creation Workflow

The following is a high-level workflow on how to configure individual network models. The detailed steps differ depending on what type of interface you use (Expert Mode, Cisco WAE UI, or Cisco WAE CLI).

If you plan to run multiple NIMOs and consolidate the information into one final network, do not run collections until after you have set up the aggregator NIMO. For more information, see NIMO Collection Consolidation, on page 62.

1. Configure device authgroups, SNMP groups, and network profile access.

2. (Optional) Configure agents. This step is required only for collecting XTC, LAG and port interface, multilayer, netflow, or telemetry information.

3. Configure an aggregated network and sources with a topology NIMO.

4. Configure additional collections such as demands, traffic, layout, inventory, and so on.

5. Schedule when to run collections.

6. Configure the archive file system location and interval at which plan files are periodically stored.

7. (Optional) View plan files in Cisco WAE applications.

CHAPTER 2

Network Model Configuration—Cisco WAE UI

This section contains the following topics:

Cisco WAE UI Overview

The Cisco WAE UI provides an easy-to-use configuration tool for device and network access, network model creation, user management, agent configuration, and so on.

For basic network model configuration we recommend starting with the Network Model Composer. You can also choose to perform certain operations using the Expert Mode or the Cisco WAE CLI. Regardless of the interface you use, the last committed configuration is saved.
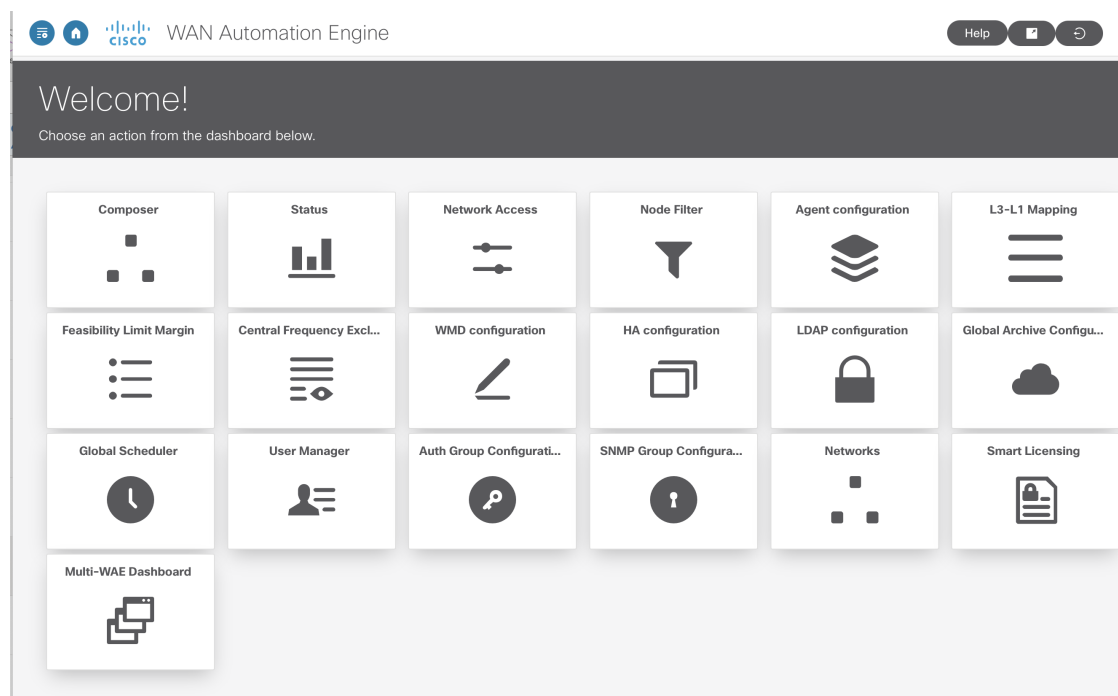
> ✎
>
> **Note** Make sure that only a single session is active per user. Multiple sessions per user can create issues and is not recommended.

Icons	Description	For more information, see...
🏠	Returns you to the main Cisco WAE UI landing page.	—
Composer	Opens the Network Model Composer, which lets you create and build a network model.	Use the Network Model Composer, on page 16
Status	The Status Dashboard helps to identify processes that cause system leaks or processes that completely use the resources.	Status Dashboard, on page 176
Network Access	Opens the network access configuration page, which lets you configure global device and network credentials.	Configure Network Access Using the Cisco WAE UI, on page 12
Node Filter	Open Node Filter page which lets you to exclude or include individual nodes from collection.	Configure the Node Filter using Cisco WAE UI, on page 15
Agent Configuration	Opens the agent configuration page, which lets you create and modify agents.	Configure Agents Using the Cisco WAE UI, on page 14
L3-L1 Mapping	Opens the L3-L1 mapping configuration page, which lets you create and modify L3-L1 node and circuit mappings for multilayer collection.	Configure L3-L1 Mapping Information, on page 97
Feasibility Limit Margin	Opens the feasibility limit margin configuration page, which lets you set the acceptable quality of the L1 circuit path.	"L1 Circuit Wavelengths" topic in the Cisco WAE Design User Guide.
Central Frequency Excludelists	Opens the central frequency excludelist configuration page, which lets you define the list of frequency IDs that may not serve as central frequency IDs for L1 circuit paths.	"L1 Circuit Wavelengths" and "Central Frequency ID Excludelist" topic in the Cisco WAE Design User Guide.
WMD configuration	Opens the WMD configuration page, which lets you view WMD options such as debugging, rpc and application subscriber connections, demands, and so on. To edit these options, use the Expert Mode or WAE CLI or WAE UI.	Cisco WAE Modeling Daemon (WMD) Configuration, on page 93
HA configuration	Opens the high availability (HA) configuration page, which lets you designate which nodes are used for HA.	Configure High Availability, on page 169

Icons	Description	For more information, see...
LDAP configuration	Opens the LDAP configuration page, which lets you enable and configure LDAP details.	• Configure LDAP, on page 173 • Configure LDAP Using the WAE UI, on page 174
Global Archive Configuration	Opens Global Archive page from where plan files can be downloaded.	Download Plan Files, on page 24
Global Scheduler	Opens Global Scheduler page which lets you to create a scheduled task.	Schedule Jobs Using the Network Model Composer, on page 23
User Manager	Opens the user management page, which lets you add, modify, and delete users.	Manage Users, on page 161
Auth Group Configuration	Opens Auth Group Configuration page which lets you create a new authorization group.	Configure Network Access Using the Cisco WAE UI, on page 12
SNMP Group Configuration	Opens SNMP Group Configuration page which lets you create a new SNMP group.	Configure Network Access Using the Cisco WAE UI, on page 12
Networks	Opens Networks page which lets you to create a standalone network.	Create a Standalone Network, on page 15
Smart Licensing	Opens Smart Licensing page that allows you to enable and register smart license for Cisco WAE.	Cisco Smart Licensing, on page 153
Multi WAE Dashboard	Opens Multi WAE Dashboard page that provides details related to different WAE instances. **Note** Multi WAE Dashboard is available only if Multi WAE is configured.	Multi WAE Collection, on page 127
	Toggles the main Cisco WAE UI navigation menu on the left (also called the left sidebar menu).	—
Help	Launches the Online Help for Cisco WAE UI in a new tab. **Note** Documents are sometimes updated after original publication. Refer to the Cisco WAE User Guide document on Cisco.com for latest updates.	—
	Launches the Expert Mode in another window.	Network Model Configuration—Expert Mode, on page 25
	Logs the current user out.	—

Configure a Network Model Using the Cisco WAE UI

This workflow describes the high-level steps to create a network model using the Cisco WAE UI and the Network Model Composer.

Step	For more information, see...
1. Configure device credentials (network authgroups and SNMP groups).	Configure Network Access Using the Cisco WAE UI, on page 12
2. (Optional) Create agents to collect specific information. Agents are needed for collecting information using XTC or for multilayer collection.	Configure Agents Using the Cisco WAE UI, on page 14
3. (Optional) Create a network which is not a DARE network	Create a Standalone Network, on page 15
Use the Network Model Composer to do the following:	
3. Create a network model and run topology collection.	Create a Network and Configure Topology Collection, on page 17
4. Configure additional data collections using NIMOs.	Configure Additional NIMOs Using the Network Model Composer, on page 18
5. Aggregate NIMOs to build a network.	Consolidate NIMO Collections Using the Network Model Composer, on page 19
6. (Optional) Configure traffic collection and customer scripts to run in your network.	Run Traffic Collection or a Custom Script Using the Network Model Composer, on page 21
8. (Optional) Configure archives.	Configure the Archive Using the Network Model Composer, on page 22
7. (Optional) Create scheduling jobs to run network collections and agents.	Schedule Jobs Using the Network Model Composer, on page 23

Configure Network Access Using the Cisco WAE UI

In this task, you are defining global device credentials by creating a network access profile.

Before you begin

Know the global network device credentials.

Procedure

Step 1   From the WAE UI, click Network Access.

Step 2   Click + **Create Network Access**.

Step 3   Enter the global device credentials:

- **Name**—Enter a name for the network access profile.
- **Login Type**—Choose which login protocol to use: SSH or Telnet. The SSH protocol is more secure. The Telnet protocol does not encrypt the username and password.
- **Authorization Group**—Choose default or create a new authorization group. If creating a new authorization group, enter a name for it and applicable information in the fields that follow.

Note	You can also create a new Authorization Group directly from WAE UI. From Cisco WAE UI, select **Auth Group Configuration** and click **Create Auth Group** . Enter the details and click **Save**.

Step 4 Choose default or create a new SNMP group. If creating a new SNMP group, enter a name for it and select either SNMPv2c or SNMPv3.

Note	You can also create a new SNMP Group directly from WAE UI. From Cisco WAE UI, select **SNMP Group Configuration** and click **Create SNMP Group**. Enter the details and click **Save**.

- If SNMPv2c, enter the SNMP RO community string that acts as a password. It is used to authenticate messages sent between the node and the seed router.
- If SNMPv3, enter the following default credentials:
    - **Security Level**—Select one of the following:
        - **noAuthNoPriv**—Security level that does not provide authentication or encryption. This level is not supported for SNMPv3.
        - **authNoPriv**—Security level that provides authentication but does not provide encryption.
        - **authPriv**—Security level that provides both authentication and encryption.
    - **Authentication Protocol and Password**—Select one of the following:
        - **md5**—HMAC-MD5-96 authentication protocol
        - **sha**—HMAC-SHA-96 authentication protocol
    - **Encryption Protocol and Password**—The priv option offers a choice of DES or 128-bit AES encryption for SNMP security encryption. The priv option and the aes-128 token indicates that this privacy password is for generating a 128-bit AES key #.The AES priv password can have a minimum of eight characters. If the passphrases are specified in clear text, you can specify a maximum of 64 characters. If you use the localized key, you can specify a maximum of 130 characters.

Step 5 Click **Save**.

Step 6 (Optional) To add or edit nodes associated with these network access credentials, do the following:

a) Click the **Edit Node Access** button and do one of the following:

- To Export a node list, click .

- To import a node list, click , and enter the CSV file path in **file-path** field and click **Done**. This overwrites nodes that were previously configured.

Note	Make sure that the CSV file is located on the server where WAE is installed.

• To add a node, click , and enter node details.

• To edit a node, select a node, click , and enter node details.

• To delete a node, select a node, click .

• To delete all nodes in bulk, click **Delete All** button.

b) Click **Done**.

Step 7   Click **Save**.

What to do next

Use the Network Model Composer to create a network model.

Configure Agents Using the Cisco WAE UI

Agents perform information-gathering tasks and should be configured before certain network collection operations. This section describes how to configure agents using the Cisco WAE UI.

Procedure

Step 1   From the Cisco WAE UI, click Agent Configuration.

Step 2   Click **Create New Agent**.

> **Note**   Telemetry and Netflow agents are created with default config. You cannot add a new telemetry or a netflow agent. However, you can change or delete the configuration by clicking on the card.
>
> To delete other agents use the trash icon.

Step 3   Enter a name for the agent.

Step 4   From the Collector Type drop-down list, select a collector.

Step 5   Click **Create Agent**.

Step 6   On the next window, enter applicable agent configuration values. To view field descriptions, hover the mouse pointer over the field name.

Step 7   Click **Save**.

Step 8   To run the agent, click **Actions** > **run-collection**.

What to do next

Use the Network Model Composer to configure NIMOs to build a network model. For more information on NIMO types, see NIMO Descriptions, on page 55.

Configure the Node Filter using Cisco WAE UI

Cisco WAE UI enables you to include or exclude individual nodes from collection.

Note
- Node name/loopback IP can be added for node filter list, management IP must not be added in node filter IPs.
- Node name works with ISIS.
- OSPF database does not have node names, so filtering only works by IP address.
- Node filter does not support Segment List hops.

Procedure

Step 1   From the Cisco WAE UI, click Node Filter.

Step 2   Click **Create Node Filter**.

Step 3   Enter the **Name** name for the Node Filter.

Step 4   Use the **Regex** option when multiple nodes are to be included/excluded with a single expression. Enter **Regex** and choose INCLUDE ONLY or EXCLUDE ONLY from **Regex Filter** dropdown.

   Note   Choose IGNORE FILTER if using **Node Filter** option.

Step 5   Alternatively, instead of **Regex**, you can add IPs or node names for each node. For this choose INCLUDE ONLY or EXCLUDE ONLY from **Node Filter** dropdown.

   Click **Add New Node** to list the nodes to be included or excluded. You can select multiple nodes using comma as a separator.

   Note   Choose IGNORE FILTER if using **Regex** option.

Step 6   Click **Save**.

Create a Standalone Network

Use the following steps to create a separate network which is not a DARE network.

Procedure

Step 1   From the Cisco WAE UI, click **Networks**.

Step 2   Click **Create New Network**.

Step 3   Enter a name for the network.

      Note   The network model name cannot be changed after it is entered.

Step 4   Click **Create Network**.

      Note   After network creation, the type of network is 'unknown'. You need to configure the network.

Step 5   Click the new network that you just created.

Step 6   Click **Choose NIMO Type** and select a NIMO from the drop down list. Click **Next**.

Step 7   Click the Collector icon to configure collection.

Step 8   Enter applicable configuration details. Hover the mouse pointer over each field to view field descriptions.

Step 9   Click **Save**. You are brought back to the main network model window.

Step 10   Click **Archive Config** to archive the configuration.

    a) Enter the **Archive Path**

    b) In the **Include NetInt Tables** field, select true or false.

    c) Enter values for cleanup action.

    d) Click **Save**.

      Note   Use different directories for different network archives. Using the same directory for multiple archives can cause loss/corruption of plan files.

Step 11   Click the Collector icon and click **Actions**.

Step 12   Click the button that will start the NIMO collection (typically "run-collection").

Use the Network Model Composer

The Network Model Composer hides the complexity of network model configuration. It provides a visual workflow to guide you from creating a network model using various NIMOs to setting up a schedule to run collections and configuring an archive to store the network model plan files.

The Network Model Composer provides the following general controls.

Add section on the right of the screen starts the process of creating a network model, an agent, a NIMO, a scheduled task, or an archive.

The numbered navigation at the top of the page displays where you are in the network model configuration process. As you complete each step, you may click on a step that you may want to skip or go back to.

The network model you are configuring - collections (NIMOs), scheduling tasks, or archives for is displayed on the left area followed by configured components (NIMOs, scheduling tasks, or archives) that have been created for the selected network model.

The configuration options for the selected component is displayed at the bottom.

Create a Network and Configure Topology Collection

The initial step in configuring a complete network model is to create a new network with topology collection. In this task, you are configuring a topology collection that will be the source network for additional network collections. After the initial collection, the node IP address table is populated and you can add management IP addresses. For more information on basic topology collections, see Basic Topology Collection, on page 58.

> **Note**   It is recommended that you configure network access and any necessary agents as described in Configure a Network Model Using the Cisco WAE UI, on page 12 before you use the Network Model Composer. However, you have the opportunity to configure network access and agents in the Network Model Composer.

Procedure

Step 1   From the Cisco WAE UI, click Network Model Composer.

Step 2   Click + **Create New Network**.

Step 3   Enter a network model name and click **Create Network**.

	Note	The network model name cannot be changed after it is entered.

Step 4   Click **Add Discovery Method**.

Step 5   Enter a **Name** for the collector and select a NIMO from the **Type** dropdown.

To select from an existing network, click **Existing network** checkbox and choose **Name** and **Type** from the dropdown.

	Note	Name of the collector must not include any space.

Step 6   Click + **Add**.

Step 7   Click the topology icon (Topo IGP or Topo BGPLS XTC or SR Traffic Matrix) to configure collection.

Step 8   Enter applicable configuration details. Hover the mouse pointer over each field to view field descriptions.

Step 9   Select **node-filter** from the dropdown. If you have not defined a node-filter, click + and enter the details. Choose between Include Filter, Exclude Filter and Ignore Filter. See Configure the Node Filter using Cisco WAE UI, on page 15

Step 10   Click **Save**. You are brought back to the main network model window.

Step 11   Click the topology icon (Topo IGP or Topo BGPLS XTC) again.

Step 12   Click **Actions** and select one of the following:

- **run collection** or **run-xtc-collection**—Starts collection.
- **Update Node List**—Allows you to delete, add, or edit existing nodes.
- **Done**—Takes you back to the previous window.

Step 13   Click **Save**.

What to do next

You can configure more collections using other NIMOs to create a complete network model.

Configure Additional NIMOs Using the Network Model Composer

This topic describes the general procedure to configure additional NIMOs using the Network Model Composer. For NIMO descriptions, see NIMO Descriptions, on page 55.

Note	• The term "Collector" is displayed in the Network Model Composer when you are prompted to enter or select a NIMO. The terms NIMO and Collector are used interchangeably in the Network Model Composer.
	• Configure additional topology NIMOs at this step. It is recommended that non-topology NIMOs (layout, inventory, demand deduction, and so on) are configured in the external-executable-nimo after topology aggregation.

Before you begin

Confirm that the network model you are working with has a basic topology NIMO configured.

Procedure

Step 1 From the Cisco WAE UI, click Network Model Composer and click a network model that you want to configure a NIMO for.

Step 2 Click **Add Discovery Method**.

Step 3 Enter a **Name** for the collector and select a NIMO from the **Type** dropdown.

To select from an existing network, click **Existing network** checkbox and choose **Name** and **Type** from the dropdown.

| Note | Name of the collector must not include any space. |

Step 4 Click + **Add**.

Step 5 Click the Collector icon to configure collection.

Step 6 Enter applicable configuration details. Hover the mouse pointer over each field to view field descriptions.

| Note | You can also refer to the Network Interface Modules (NIMOs), on page 55 topic. This topic links to NIMOs and associated configuration options. |

Step 7 Click **Save**. You are brought back to the main network model window.

Step 8 Click the Collector icon and click **Actions**.

Step 9 Click the button that will start the NIMO collection (typically "run-collection").

What to do next

You can do the following:

- Configure and run more collections

- Consolidate NIMO Collections Using the Network Model Composer

Consolidate NIMO Collections Using the Network Model Composer

After you configure multiple NIMOs, you will want to consolidate all the collection models to build a complete network model. After NIMO aggregation, you can collect traffic statistics (traffic-poll-nimo) and run custom scripts (external-executable-nimo) against your network model.

Figure 1: Network Model Composer—Aggregation
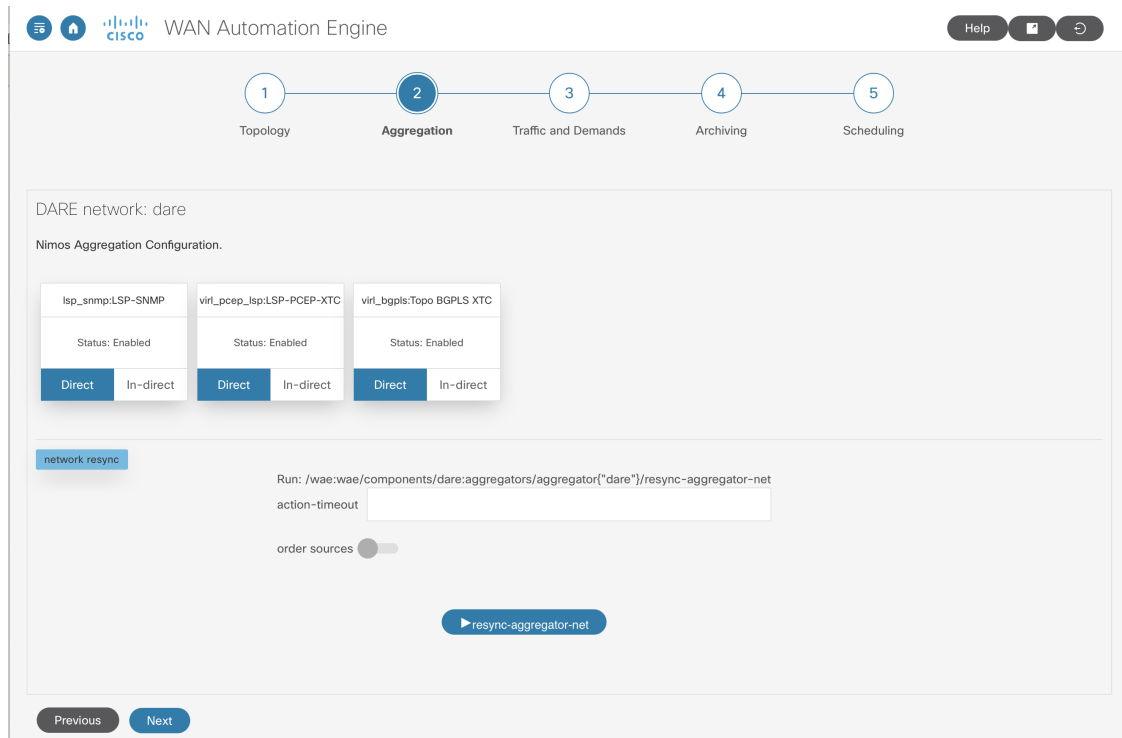
Procedure

Step 1   Click Network Model Composer and choose a network model.

Step 2   Click the **Aggregation** icon from the top navigation bar. The Aggregation page should look similar to the one above.

Step 3   By default, all NIMOs are included in the aggregation. To exclude any NIMOs from aggregation, click **Indirect**. Any changes on that collection model will not be included during aggregation.

Step 4   In the **network rebuild** section, you can choose to modify the order of the sources. Enable **order source** and use the arrows to change the order from the **Ordered Sources** list. For more information on aggregation, refer to the NIMO Collection Consolidation, on page 62 topic.

Run Traffic Collection or a Custom Script Using the Network Model Composer

Figure 2: Network Model Composer—Traffic and Demands

Before you begin

Confirm you have completed the preliminary tasks described in Configure a Network Model Using the Cisco WAE UI, on page 12 and have aggregated collection models.

Procedure

Step 1   Click Network Model Composer and choose a network model.

Step 2   Click **Traffic and Demands**.

Step 3   (Optional) In the **Final Network** field, type a name to configure the final SAGE network and click **Save**.

Leave the **Final Network** field blank if you do not want to configure a SAGE network.

> **Note**   Click **reset-final-network** to reset the final network.

Step 4   Click **Add Collector**.

Step 5   Enter a **Name** for the collector and select a NIMO from the **Type** dropdown.

To select from an existing network, click **Existing network** checkbox and choose **Name** and **Type** from the dropdown.

> **Note**   Name of the collector must not include any space.

Step 6   Click **Add**. The new collector appears.

Step 7   Click the collector you just created.

Step 8   Enter applicable configuration details. Hover the mouse pointer over each field to view field descriptions.

Step 9   Click **Save**. You are brought back to the main network model window.

Configure the Archive Using the Network Model Composer

After creating a network model and running collections, you have the option to retrieve and view plan files. Plan files capture all relevant information about a network at a given time, and can include topology, traffic, routing, and related information.

The Archive is a repository for plan files. See also Configure the Archive Using the WAE CLI, on page 52, which describes how to configure the Archive using the Cisco WAE CLI.

> **Note**   To schedule archiving, see Schedule Jobs Using the Network Model Composer, on page 23

Figure 3: Network Model Composer - Archiving
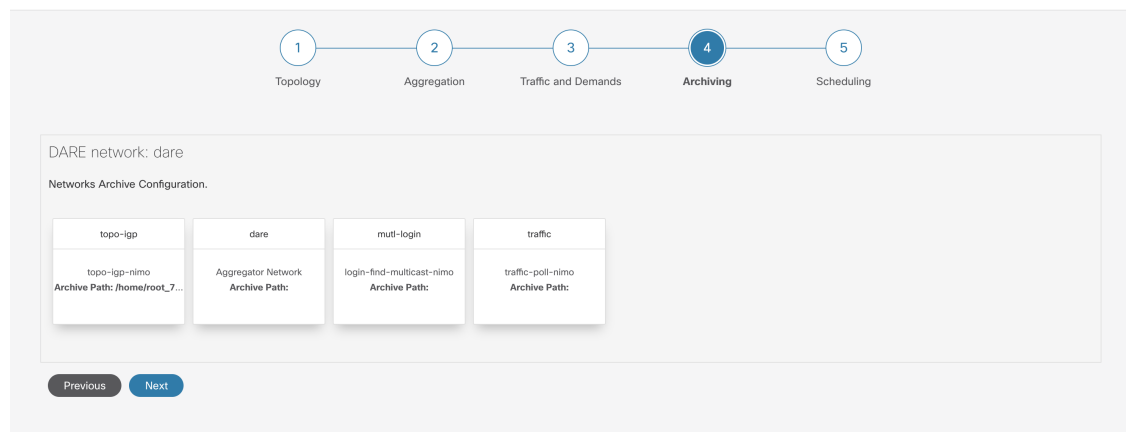
Procedure

Step 1   Click Network Model Composer and choose a network model.

Step 2   Click **Archiving**.

Step 3   Click a NIMO or the network model you want to configure the archive for.

Step 4   The Archive path might be populated if they were configured when the network was initially created. If not, or if you want to change them, enter new values.

Step 5   Select the source that the Archive will retrieve from.

Step 6   In the Cleanup section, select a value for **Enable** field and enter **Retain Number of Days** value.

Step 7   Click **Save**.

Step 8   Click **Enable** to enable the archive.

Note   Archiving must be configured first before it is enabled.

Schedule Jobs Using the Network Model Composer

This procedure describes how to schedule different network collections and agents to run using the Network Model Composer. For more information on additional scheduling jobs and configuration options that can be configured using the Expert Mode, see .

Figure 4: Network Model Composer - Scheduling
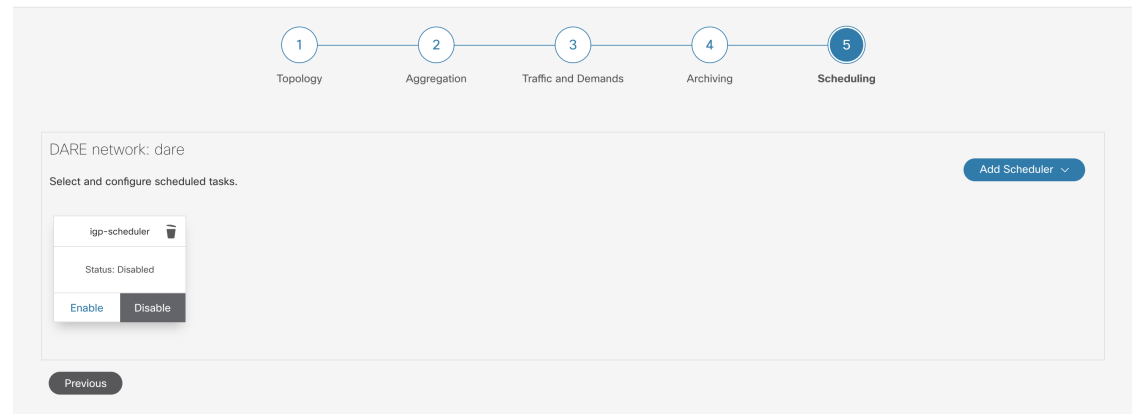
Procedure

Step 1	Click Network Model Composer and choose a network model.
Step 2	Click **Scheduling**.
	You can also create a scheduled task directly from WAE UI. From Cisco WAE UI, select **Global Scheduler** and click + **Create Scheduled Task**. Enter a name for the scheduler and click **Create Scheduled Task**. Click the new scheduler you just created and enter the necessary details.
	The **Global Scheduler** page lists all the schedulers that are configured irrespective of the network.
Step 3	Click **Add Scheduler**.
Step 4	Enter a name for the scheduling job.
Step 5	Click **Add**.
Step 6	Click the scheduling job icon.
Step 7	To add an action, click [icon] and enter an action name.
Step 8	Enter the order number in the **order** field.
Step 9	Choose whether the action will be performed on a NIMO or agent or an aggregated network.
Step 10	Select the NIMO or agent or Aggregated network from the drop-down list.
Step 11	If the action-type and path fields are not populated, enter applicable values.

Step 12	Click **Save**.
Step 13	(Optional) Add more actions.

Step 14   To add a trigger, click and enter a trigger name.

Step 15   Select **Every**. In the **Run Every** field, enter the time interval and select the appropriate unit of time.

If you are familiar with cron configuration, select **Cron Expression** and configure the time interval to run the actions.

Step 16   Click **Save**.

Step 17   Click **Run Task**.

Note	Each action is done in the order it is listed and configured.

Download Plan Files

The network model is saved in a plan file (.pln format) which can be downloaded.

Before you begin

Make sure that archive has been configured.

Procedure

Step 1   From the Cisco WAE UI, click **Global Archive Configuration**.

Step 2   Use **TimeZone** option to change the default value to your local time zone. For example, +530, -7, etc. Click **Add Time Zone**.

Note	This option only changes the way the plan files are displayed in the calendar in GUI. It does not change the timezone of the system or the file names.

Step 3   Click the network that you have configured for archiving.

Step 4   Month view of the calendar opens. Click the date for which you intend to download the plan file.

Step 5   Day view opens along with the list of .pln files that are archived.

Step 6   Click the .pln file to download the file.

CHAPTER 3

Network Model Configuration—Expert Mode

This section contains the following topics:

Expert Mode Overview

The Expert Mode browser has additional device and service functionality that might not be available in the WAE UI. You might also prefer to use the Expert Mode over the WAE CLI because all options for each operation are visible on the Expert Mode.

The Expert Mode is a mix of custom-built widgets and auto-rendering from the underlying device, service, and network models. The Expert Mode is immediately updated when new devices, NIMOs, or network models are added to the system.

In the top-right corner of the Cisco WAE UI (https://server-ip:8443), click the icon to access the Expert Mode:

The purpose of this section is to illustrate the Expert Mode and go over procedures to get you up and running. This section does not go into advanced configurations. It is assumed that once you understand the basic procedures, you can configure more complex operations.
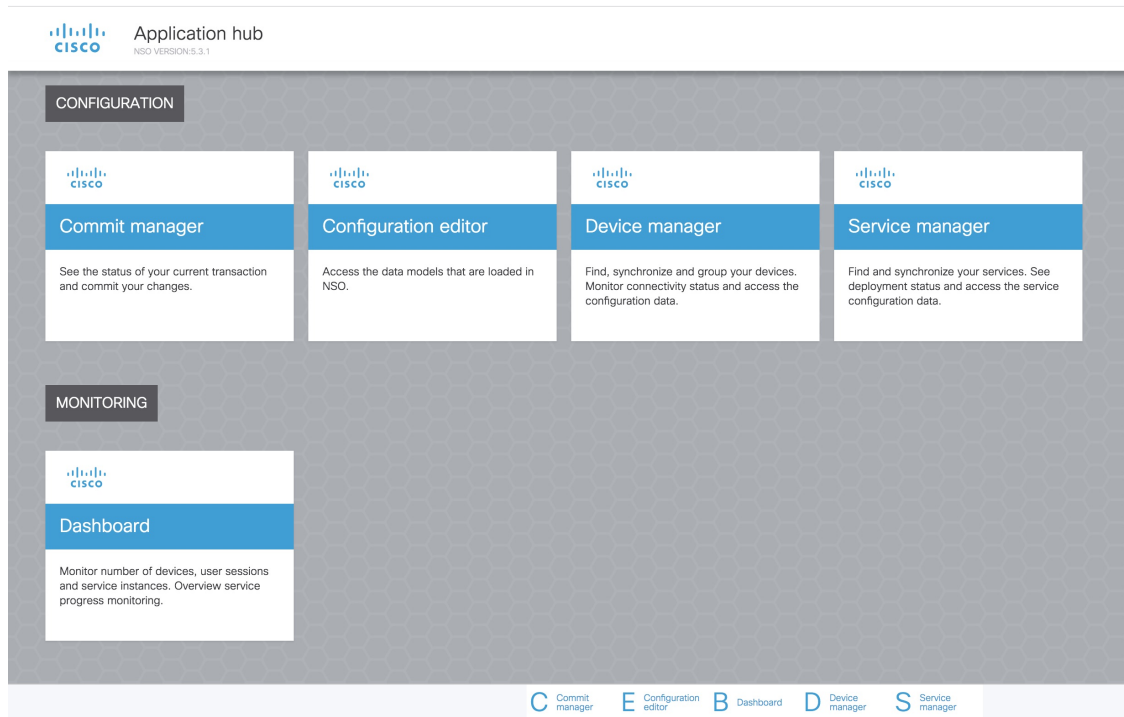
Use the **Configuration editor** to access the data models. Under **Modules** tab, click:

- wae:wae - to configure global settings and agents.

- wae:networks - to configure network settings and NIMOs.

Use **Commit manager** to commit any changes. Click the **Commit** button to save the configuration changes made.

Navigation and Commit

When an object type (for example, a network instance) is selected, a list of all related object instances is shown. When performing network configuration operations, navigate to **Commit manager** and click **Commit** button to save the changes. For more information on Commit functionality, see Commit Flags, on page 195.

Configure a Network Model Using the Expert Mode

This workflow describes the high-level configuration steps on how to create a network model using the Expert Mode.

Step	For more information, see...
1. Configure device authgroups and SNMP groups.	Configure Device Access Using the Expert Mode, on page 27
2. Configure a network access profile.	Configure Network Access, on page 28

Step	For more information, see...
3. Configure agents. **Note** This step is only required for collecting XTC or multi-layer information.	• Configuring XTC Agents Using the Expert Mode, on page 28 • Configure the Configuration Parsing Agent Using the Expert Mode, on page 32
4. Create a network and collect basic topology data. **Note** If you plan to consolidate network models and collect more than basic topology (for example, merge topo-bgpls-xtc-nimo and lsp-pcep-xtc-nimo information into one final network model), configure DARE and create networks where collections have not yet been executed. For more information, see NIMO Collection Consolidation, on page 62.	Create a Network Model, on page 33
5. Configure additional data collections.	Configure Additional NIMOs, on page 34
6. (Optional) Configure the Scheduler.	Scheduler Configuration, on page 149
7. Configure and view plan archives.	Configure the Archive and View Plan Files Using the WAE Expert Mode, on page 34

Configure Device Access Using the Expert Mode

Cisco WAE uses authgroups for login and SNMP access to the devices. The following procedure describes how to use the Expert Mode to configure authgroups and network access.

Procedure

Step 1 From the Expert Mode, set up authgroups.

    a) In **Configuration editor**, navigate to **/ncs:devices** and click the **authgroups** tab.

    b) Click **group**.

    c) Click the plus (+) sign, enter an authgroup name, and click **Add**.

    d) Click **default-map** and enter the default authentication parameters. For example, choose **remote-name** from the drop-down list, check the **default-map** check box, and enter the remote name string credentials.

        Note If not visible, scroll down to view and enter the remote secondary password.

Step 2 Set up SNMP groups.

    a) Navigate back to **/ncs:devices/authgroups** and click the **snmp-group** tab.

    b) Click the plus (+) sign and enter an SNMP group name.

    c) Check **default-map** and enter the default SNMP credentials. For example, choose **community-name** from the drop-down list, check the **default-map** check box, and enter the community string credentials.

    d) If configuring SNMPv3, click the **usm** tab and enter in the applicable User-based Security Model (USM) values (remote user, security level, authentication, and privacy protocols). For more information on USM values, see the

SNMPv3 options explained in the following WAE UI procedural topic: Configure Network Access Using the Cisco WAE UI, on page 12.

e) Navigate to **Commit manager** and click **Commit** button.

What to do next

Create a network access profile. See Configure Network Access, on page 28.

Configure Network Access

Procedure

Step 1   In **Configuration editor**, navigate to **/wae:wae** and click the **nimos** tab.

Step 2   Click **network-access**.

Step 3   Click the plus (+) sign and enter a network access name.

Step 4   Select and enter the appropriate network access details.

The default authgroups and SNMP groups you configured earlier are available in the drop-down lists. For more information, see Configure Device Access Using the Expert Mode, on page 27.

Step 5   Click the **node-access** tab to enter the management IP address of the router.

a) Click the plus (+) sign, enter the IP address, and click **Add**.

b) Enter the associated management IP.

Repeat these steps as necessary for all management IPs.

Step 6   Navigate to **Commit manager** and click **Commit** button.

What to do next

After completion of this task, you can create a network and run basic data collection.

Configure Agents Using the Expert Mode

Agents perform information-gathering tasks and must be configured before certain network collection operations. This section describes how to configure these agents using the Expert Mode.

Configuring XTC Agents Using the Expert Mode

The XR Transport Controller (XTC) agent periodically collects information from XTC and keeps it as raw and normalized data. The agent is used to connect to the REST interface of XTC and retrieve the PCE topology. This data is consumed by different applications (Bandwidth on Demand) and NIMOs (topo-bgpls-xtc-nimo and lsp-pcep-xtc-nimo) to extract topology, LSPs, and so on. An agent must be configured for every XTC node in the network. XTC agents must be configured for any networks that use XTC before you can perform a network collection.

Procedure

Step 1   From the Expert Mode, in **Configuration editor**, navigate to **wae:wae** and click the **agents** tab.

Step 2   Click **xtc**.

Step 3   Click the plus (+) icon to add agents.

Step 4   Enter the following information:

- XTC agent name.
- **xtc-host-ip**—Host IP address of the XTC router.
- **xtc-rest-port**—Port number to use for REST calls to the XTC host. The default is 8080.
- **use-auth**—To use HTTP basic authentication with defined credentials, choose **true** from the drop-down list.
- **auth-group**—XTC credentials that were defined in Configure Device Access Using the Expert Mode, on page 27.
- **batch-size**—Number of nodes to send in each message. Default is 1000.
- **keep-alive**—Interval in seconds to send keep-alive messages. Default is 10.
- **max-lsp-history**—Number of LSP entries to send. Default is 0.
- **enabled**—Enables the XTC agent. Default is true.

As long as the **enabled** option is set to true, the XTC agent starts right away after configuration or when WAE starts. In the same respect, the XTC agent stops when the configuration is removed, if WAE has stopped, or the enabled option is set to false.

Step 5   Click **Commit**.

Step 6   Repeat these steps for all XTC nodes.

Step 7   To view the raw data, navigate back to **/wae:wae/agents/xtc-agent:xtc/xtc/<*agent-name*>** and click the **pce** tab.

Step 8   Click the appropriate data containers (topology-nodes, tunnel-detail-infos, and xtc-topology-objects) to view the raw data.

To confirm data was collected successfully, navigate to **/wae:wae/agents/xtc-agent:xtc/xtc/<*agent-name*>** and click the **status** tab. You can view the time stamp of the last successful collection.

Note   The WAE XTC agent keeps updating in the back end even after a restart is successful. The status of XTC agent updation can be seen from data received and data reports flags from XTC agent status. The value of more than 0, means that agent update is completed.

What to do next

Configure collections for networks that use XTC. For more information, see NIMO Descriptions, on page 55.

Adding Buffer Time

You can add buffer time to process notifications in an XTC agent. If buffer is enabled, the XTC agent processes the notification, and only after the buffered time, the consolidated notification is sent to topo-bgpls-xtc-nimo and lsp-pcep-xtc-nimo. This feature is helpful if there are too many back to back notifications like link flapping, etc.

The XTC agent can be configured to collect only Topology information or LSP information using the below configuration.

```
wae agents xtc xtc xtc-agent-name advanced topology-collection
```

Options:

off	XTC agent does not collect the topology details
collection-only	XTC agent collects the topology information but does not subscribe for notification
collection-and-subscription	XTC agent collects and receives notification updates from SR PCE server

Similarly we can configure XTC agent for LSP collection and notifcation using lsp-collection advanced attribute.

```
wae agents xtc xtc agent_name advanced
```

events-buffer-time	Time to buffer XTC events before sending to NIMOs, in seconds (0 = disabled).
lsp-collection	Whether to collect LSP data and to have subscription for network changes.
topology-collection	Whether to collect topology data and to have subscription for network changes.

Configuring Netflow Agents Using the Expert Mode

The netflow agent collects and aggregates exported NetFlow and related flow measurements. These measurements can be used to construct accurate demand traffic data for WAE Design.

Procedure

Step 1   From the Expert Mode, in **Configuration editor**, navigate to **wae:wae** and click the **agents** tab.

Step 2   Click **netflow**.

Step 3   Select the operation mode from **mode** drop down list. Choose between **single** or **controller and processor**. Select single mode for CNF collection.

Step 4   In the **config** tab, enter all the details for **controller**, **processor**, **jms** and **common**.

The controller node, processor node, jms and common folders are displayed based on the operation mode selected.

Note   For CNF, though the mode is Single, it is mandatory to configure the controller node, processor node. Give valid json file for `cluster-config-file-path` under Controller tab, and valid `service-instance-id` under Processor tab matching the entry inside json file.

Step 5   Click **Commit**.

Step 6   In the **netflow** tab, click **start** > **Invoke start** to start collecting.

Step 7   Click **stop** > **Invoke stop** to stop collecting.

Step 8   Run the **status** > **Invoke status** to request agent status using CLI:

```
wae agents netflow status
status true
message
CLUSTER STATUS - BEGIN

    AGENT NODE - BEGIN
        cluster ID:                    wae-netflow-agent
        instance ID:                   agent-single
```

Note   • Open standard OS terminal.

• Ensure `waerc` file is sourced.

• Run following command:

`sudo /home/wae/test/run/packages/cisco-wae-netflow-agent/priv/bin/flow_cluster_manage`
`-action prepare-os-for-netflow`

• Stop wae and restart your system.

You need to do this only once before the first run of netflow collection.

Example

Following is a sample agent config:

```
admin@wae# show running-config wae agents netflow
wae agents netflow mode single
wae agents netflow config controller cluster-config-file-path
/opt/wae-netflow/flow-config-single.json
wae agents netflow config processor service-instance-id agent-single
wae agents netflow config common log-level debug
```

What to do next

Configure collections for networks that use Netflow. For more information, see .

Configuring SNMP Agents using the Expert Mode

The SNMP agent can be configured to collect data using continuous poller to build a network model.

Procedure

Step 1   From the Expert Mode, n **Configuration editor**, navigate to **wae:wae** and click the **agents** tab.

Step 2   Click **snmp**.

Step 3   Click the plus (+) icon to add agents.

Step 4   Enter the name and click **confirm**.

Step 5   Click the poller that you created and enter the details:

- Set **network-access** using the dropdown.

- Set **enabled** to true.

Step 6   Click **discovery** tab. Enable and set polling periods for **node-discovery** and **interface-discovery**.

Step 7   Click **run-snmp-poller**.

Configure the Configuration Parsing Agent Using the Expert Mode

The Configuration Parse agent can collect (run-config-get) router configuration files from Cisco, Juniper, and Huawei routers .The agent can retrieve the configuration by determining the router type/vendor. After collecting the configuration files , the user can configure a cfg-parse-nimo to collect or parse LSP, VPN, port, and Shared Risk Link Groups (SRLG) data. For information on what types of router configurations the agent can read, see Router Configuration Information, on page 32.

The following procedure describes how to configure the agent using the Expert Mode. You can also use the Cisco WAE UI to configure this agent (Cisco WAE UI Overview, on page 9).

Procedure

Step 1   From the Expert Mode, navigate to **wae:wae** and click the **agents** tab.

Step 2   Click **cfg-parse**.

Step 3   Click the plus (+) icon to add agents and enter a Configuration Parsing agent name. This can be any arbitrary name.

Step 4   Click the **get** tab and choose the network access.

Step 5   Click **Commit**.

Step 6   Navigate back to the **cfg-parse** tab and click **run-config-get** > **Invoke run-config-get**.

Note   The configurations are saved at the location:

```
<wae-run-dir>/data/agents/cfg-parse/<agent-name>
```

Example:

```
admin@wae# show running-config wae agents cfg-parse cfg-parse pc-test
wae agents cfg-parse cfg-parse pc-test
 source-network topo-network
 network-access test-net-access
!
```

Router Configuration Information

The following router configuration information can be read by the Configuration Parse agent:

• Router name	• IGP type and metrics (IS-IS or OSPF)
• Router IP address (loopback)	• RSVP reservable bandwidth (MPLS)
• Interface names (inside IGP topology)	• LAG ports (for Ethernet) and bundle ports (for different link types)
• Interface IP addresses	• LSPs and LSP paths
• Interface capacities (if available)	• Named paths and named path hops
	• Segment lists and segment list hops

Create a Network Model

When creating a network you must also configure basic topology collection using the topo-igp-nimo or the topo-bgpls-xtc-nimo. The following procedure describes the first configuration step using the Expert Mode.

Before you begin

- Confirm that device access and network access are configured. For more information, see Configure Device Access Using the Expert Mode, on page 27 and Configure Network Access, on page 28.

- If creating a network running XTC, confirm that XTC agents have been configured. For more information, see Configuring XTC Agents Using the Expert Mode, on page 28.

Procedure

Step 1   From the Expert Mode, in **Configuration editor**, navigate to **/wae:networks**.

Step 2   Click the plus (+) sign and enter a network model name.

Step 3   Click **Add**.

Step 4   Click the **nimo** tab.

Step 5   From the **Choice - nimo-type** drop-down list, choose one of the following:

- **topo-igp-nimo**—Collect topology information using the IGP database. For more information on options, see Topology Collection Using the IGP Database, on page 58.
- **topo-bgpls-xtc-nimo**—Collect topology information from a network running XTC. This NIMO requires a configured agent. For more information, see Topology Collection Using XTC, on page 59 and Configuring XTC Agents Using the Expert Mode, on page 28.

Step 6   Click the corresponding link. For example, if you chose topo-igp-nimo, click the **topo-igp-nimo** link and enter the applicable parameters.

Step 7   Navigate to **Commit manager** and click **Commit** button. This network model can now be used as the source network for additional network collections.

What to do next

Configure additional network collections using this network model as the source network. For more information, see NIMO Descriptions, on page 55.

Configure Additional NIMOs

This topic only describes the general steps on configuring different types of advanced network data collection. NIMOs are used to collect different types of data. Some NIMOs require the configuration of agents. For more information, see NIMO Descriptions, on page 55.

Before you begin

You must have a network model with basic collection to be used as a source network. For more information, see Create a Network Model, on page 33.

Procedure

Step 1 From the Expert Mode, in **Configuration editor**, navigate to **/wae:networks/network/***network_name*.

Step 2 Click the **nimo** tab.

Step 3 From the **Choice - nimo-type** drop-down list, select a NIMO to configure.

Step 4 Enter the appropriate parameters for the selected NIMO.

Step 5 Navigate to **Commit manager** and click **Commit** button.

Step 6 Click **run-collection > Invoke run-collection**.

Configure the Archive and View Plan Files Using the WAE Expert Mode

After creating a network model and running collection, you have the option to retrieve and view plan files. Plan files capture all relevant information about a network at a given time, and can include topology, traffic, routing, and related information.

The Archive is a repository for plan files. See also Configure the Archive Using the WAE CLI, on page 52, which describes how to configure the Archive using the WAE CLI.

Procedure

Step 1 From Expert Mode, in **Configuration editor**, navigate to **/wae:networks/network/***<network_model_name>* and click the **plan-archive** tab.

Step 2 Enter the archive directory.

Step 3 Navigate to **Commit manager** and click **commit**.

Step 4 To save the current network model to a plan file into the archive directory you specified, click **run**.

Step 5   To retrieve a plan file:

a) Click **get**.

b) Enter timestamp and plan format.

Hover over the fields to view what format is expected.

c) Click **Invoke get**.

The output obtained is encoded string. Open the file from WAE Design GUI.

What to do next

From the WAE Design GUI, you can open plan files residing on WAE Archive.

Network Model Configuration—Cisco WAE CLI

This section contains the following topics:

WAE CLI Overview

WAE provides a network CLI that is automatically rendered using the data models described by the WAE YANG files. The CLI contains commands for manipulating the network configuration. The CLI is entirely data-model driven. The YANG model(s) define a hierarchy of configuration elements; the CLI follows this tree. The CLI provides various commands for configuring hardware and network connectivity of managed devices.

The CLI supports two modes: *operational mode*, for monitoring the state of WAE nodes; and *configure mode*, for changing the state of the network. The prompt indicates which mode the CLI is in. When moving from operational mode to configure mode using the **configure** command, the prompt is changed from **user@wae#** to **user@wae(config)#**. The prompts can be configured using the c-prompt1 and c-prompt2 settings in the wae.conf file.

For example:

```
admin@wae# configure
Entering configuration mode terminal
admin@wae(config)#
```

Operational Mode

Operational mode is the initial mode after successful login to the CLI. It is primarily used for viewing the system status, controlling the CLI environment, monitoring and troubleshooting network connectivity, and initiating the configure mode.

The following commands are the base commands available in operational mode. Additional commands are rendered from the loaded YANG files.

Invoke an action:

```
<path> <parameters>
```

Invokes the action found at *path* using the supplied *parameters*. This command is auto-generated from the YANG file. For example, given the following action specification in a YANG file:

```
tailf:action shutdown {
  tailf:actionpoint actions;
  input {
    tailf:constant-leaf flags {
      type uint64 {
        range "1 .. max";
      }
      tailf:constant-value 42;
    }
    leaf timeout {
      type xs:duration;
      default PT60S;
    }
    leaf message {
      type string;
    }
    container options {
      leaf rebootAfterShutdown {
        type boolean;
        default false;
      }
      leaf forceFsckAfterReboot {
        type boolean;
        default false;
      }
      leaf powerOffAfterShutdown {
        type boolean;
        default true;
      }
    }
  }
}
```

The action can be invoked in the following way:

```
user@wae> shutdown timeout 10s message reboot options { \
forceFsckAfterReboot true }
```

Built-in Operational Mode Commands

Command	Description
commit (abort│confirm)	Terminates or confirms a pending confirming commit. A pending confirming commit is terminated if the CLI session is canceled without doing **commit confirm**. The default is **confirm**. Example: `user@wae# ` **commit abort**

`config (exclusive`|`terminal) [no-confirm]`	Enters configure mode. The default is **`terminal`**. • terminal—Edits a private copy of the running configuration; no lock is taken. • no-confirm—Enters configure mode, ignoring any confirm dialog. Example: ``` user@wae# config terminal Entering configuration mode terminal ```
`file list `*<directory>*	Lists files in a directory. Example: ``` user@wae# file list /config rollback10001 rollback10002 rollback10003 rollback10004 rollback10005 ```
`file show `*<file>*	Displays contents of a file. Example: ``` user@wae# file show /etc/skel/.bash_profile # /etc/skel/.bash_profile # This file is sourced by bash for login shells. The following line # runs our .bashrc and is recommended by the bash info pages. [[-f ~/.bashrc]] && . ~/.bashrc ```
`help `*<command>*	Displays help text for a command. Example: ``` user@wae# help job Help for command: job Job operations ```
`job stop `*<job id>*	Stops a specific background job. In the default CLI, the only command that creates background jobs is **`monitor start`**. Example: ``` user@wae# monitor start /var/log/messages [ok][...] admin@ncs# show jobs JOB COMMAND 3 monitor start /var/log/messages admin@ncs# job stop 3 admin@ncs# show jobs JOB COMMAND ```
`logout session `*<session ID>*	Logs out a specific user session from WAE. If the user holds the **`configure exclusive`** lock, the lock is released. Example: ``` user@wae# who Session User Context From Proto Date Mode 25 oper cli 192.0.2.254 ssh 12:10:40 operational *24 admin cli 192.0.2.254 ssh 12:05:50 operational user@wae# logout session 25 user@wae# who Session User Context From Proto Date Mode *24 admin cli 192.0.2.254 ssh 12:05:50 operational ```

`logout user` *<username>*	Logs out a specific user from WAE. If the user holds the **`configure exclusive`** lock, the lock is released. Example: ``` user@wae# who Session User Context From Proto Date Mode 25 oper cli 192.0.2.254 ssh 12:10:40 operational *24 admin cli 192.0.2.254 ssh 12:05:50 operational user@wae# logout user oper user@wae# who Session User Context From Proto Date Mode *24 admin cli 192.0.2.254 ssh 12:05:50 operational ```
`script reload`	Reloads scripts found in the scripts/command directory. New scripts are added. If a script file has been removed, the corresponding CLI command is purged.
`send (all` | *<user>*`)` *<message>*	Displays a message on the screens of all users who are logged in to the device or on a specific screen. • all—Display the message to all currently logged in users. • *<user>*—Display the message to a specific user. Example: ``` user@wae# send oper "I will reboot system in 5 minutes." ``` The oper user sees the following message onscreen: ``` oper@wae# Message from user@wae at 13:16:41... I will reboot system in 5 minutes. EOF ```
`show cli`	Displays CLI properties. Example: ``` user@wae# show cli autowizard false complete-on-space true display-level 99999999 history 100 idle-timeout 1800 ignore-leading-space false output-file terminal paginate true prompt1 \h\M# prompt2 \h(\m)# screen-length 71 screen-width 80 service prompt config true show-defaults false terminal xterm-256color timestamp disable ```

`show history` [*<limit>*]	Displays CLI command history. By default the last 100 commands are listed. The size of the history list is configured using the history CLI setting. If a history limit is specified, only the last commands up to that limit are shown. Example: ``` user@wae# show history 06-19 14:34:02 -- ping router 06-20 14:42:35 -- show running-config 06-20 14:42:37 -- who 06-20 14:42:40 -- show history user@wae# show history 3 14:42:37 -- who 14:42:40 -- show history 14:42:46 -- show history 3 ```
`show jobs`	Displays jobs that are currently running in the background. Example: ``` user@wae# show jobs JOB COMMAND 3 monitor start /var/log/messages ```
`show log` *<file>*	Displays contents of a log file. Example: ``` user@wae# show log messages ```
`show parser dump` *<command prefix>*	Shows all possible commands that start with the specified command prefix.
`show running-config` [*<path filter>* [`sort-by` *<idx>*]]	Displays the current configuration. By default the entire configuration is displayed. You can limit what is shown by supplying a path filter. The path filter can be either a path pointing to a specific instance, or if an instance ID is omitted, the part following the omitted instance is treated as a filter. The `sort-by` argument can be used when the path filter points to a list element with secondary indexes. The name of a secondary index is *idx*. When used, the table is sorted in the order defined by the secondary index. This lets you control the order in which to display instances. For example, to show the aaa settings for the admin user: ``` user@wae# show running-config aaa authentication users user admin aaa authentication users user admin uid 1000 gid 1000 password 1JA.1O3Tx$Zt1ycpnMlg1bVMqM/zSZ7/ ssh_keydir /var/ncs/homes/admin/.ssh homedir /var/ncs/homes/admin ! ``` To show all users who have group ID 1000, omit the user ID and instead specify gid 1000: ``` user@wae# show running-config aaa authentication users user * gid 1000 ... ```

show *<path>* [**sort-by** *<idx>*]	Shows the configuration as a table provided that path leads to a list element and the data can be rendered as a table (that is, the table fits on the screen). You can also force table formatting of a list by using the | **tab** pipe command.
	The **sort-by** argument can be used when the path points to a list element with secondary indexes. The name of a secondary index is *idx*. When used, the table is sorted in the order defined by the secondary index. This lets you control the order in which to display instances. Example:
	``` user@wae# show devices device-module NAME REVISION URI DEVICES --- junos - http://xml.juniper.net/xnm/1.1/xnm [pe2] tailf-ned-cisco-ios - urn:ios [ce1 ce0] tailf-ned-cisco-ios-stats - urn:ios-stats [ce1 ce0] tailf-ned-cisco-ios-xr - http://tail-f.com/ned/cisco-ios-xr [p1 p0] ```
source *<file>*	Runs commands from a specified file as if they had been entered by the user. The autowizard is disabled when executing commands from the file.
timecmd *<command>*	Measures and displays the execution time of a command. Note that **timecmd** is only available if devtools has been set to true in the CLI session settings. Example:
	``` user@wae# timecmd id user = admin(501), gid=20, groups=admin, gids=12,20,33,61,79,80,81,98,100 Command executed in 0.00 sec user@wae# ```
who	Displays currently logged on users. The current session—the session running the **show status** command—is marked with an asterisk. Example:
	``` user@wae# who Session User Context From Proto Date Mode 25 oper cli 192.0.2.254 ssh 12:10:40 operational *24 admin cli 192.0.2.254 ssh 12:05:50 operational admin@ncs# ```

Configure Mode

Configure mode can be initiated by entering the configure command in operational mode. All changes to the network configuration are done to a copy of the active configuration. These changes do not take effect until a successful commit or commit confirm command is entered.

The following commands are the base commands available in configure mode. Additional commands are rendered from the loaded YANG files.

Configure a value:

```
<path> [<value>]
```

Set a parameter. If a new identifier is created and autowizard is enabled, the CLI prompts the user for all mandatory sub-elements of that identifier. This command is auto-generated from the YANG file.

If no *<value>* is provided, the CLI prompts the user for the value. No echo of the entered value occurs if *<path>* is an encrypted value of the type MD5DigestString, DESDigestString, DES3CBCEncryptedString, or AESCFB128EncryptedString as documented in the tailf-common.yang data-model.

Built-in Configure Mode Commands

Command	Description
annotate *<statement>* *<text>*	Associates an annotation with a given configuration. To remove an annotation, leave the text empty. This command is only available when the system has been configured with attributes enabled.
commit (**check** | **and-quit** | **confirmed** | **to-startup**) [**comment** *<text>*] [**label** *<text>*]	Commits the current configuration to running. • check—Validates the current configuration. • and-quit—Commits to running and quits configure mode. • comment *<text>*—Associates a comment with the commit. The comment is visible when examining rollback files. • label <text>—Associates a label with the commit. The label is visible when examining rollback files. **Note** A useful command is `commit dry-run`. This command validates and displays the configuration changes, but does not perform the actual commit. For more available `commit` commands, see Commit Flags, on page 195.
copy *<instance path>* *<new id>*	Makes a copy of an instance.

copy cfg [**merge** | **overwrite**] *<src path>* **to** *<dest path>*	Copies data from one configuration tree to another. Only data that makes sense at the destination is copied. No error message is generated for data that cannot be copied and the operation can fail completely without any error messages being generated. For example, to create a template from a part of a device config, first configure the device and then copy the config to the template configuration tree. Example:
	
	`user@wae(config)# devices template host_temp`
	`user@wae(config-template-host_temp)# exit`
	`user@wae(config)# copy cfg merge devices device ce0 config \`
	`  ios:ethernet to devices template host_temp config ios:ethernet`
	`user@wae(config)# show configuration diff`
	`+devices template host_temp`
	`+ config`
	`+ ios:ethernet cfm global`
	`+ !`
	`+!`

copy compare *<src path>* **to** *<dest path>*	Compares two arbitrary configuration trees. Items that appear only in the source tree are ignored.
delete *<path>*	Deletes a data element.
do *<command>*	Runs the command in operational mode.

edit *<path>*	Edits a sub-element. Missing elements in the path are created.
exit (**level** | **configuration-mode**)	• level—Exits from this level. If performed at the top level, exits configure mode. This is the default if no option is given. • configuration mode—Exits from configuration mode regardless of the edit level.
help *<command>*	Shows help text for the command.
hide *<hide-group>*	Rehides the elements and actions that belong to the hide groups. No password is required for hiding. This command is hidden and is not shown during command completion.
insert *<path>*	Inserts a new element. If the element already exists and has the indexedView option set in the data model, the old element is renamed as element+1 and the new element is inserted in its place.
insert *<path>* [**first** | **last** | **before** *<key>* | **after** *<key>*]	Injects a new element into an ordered list. The element can be added first, last (the default), before, or after another element.

load (**merge** | **override**) (**terminal** | *<file>*)	Loads the configuration from a file or terminal.
	
	• merge—Merges the content of the file or terminal with the current configuration.
	
	• override—Replaces the current configuration with the configuration from the file or terminal.
	
	For example, with the following current configuration:
	
	``
	devices device p1
	 config
	  cisco-ios-xr:interface GigabitEthernet 0/0/0/0
	   shutdown
	  exit
	  cisco-ios-xr:interface GigabitEthernet 0/0/0/1
	   shutdown
	 !
	!
	``
	The **shutdown** value for the entry GigabitEthernet 0/0/0/0 is deleted. Because the configuration file is just a sequence of commands with comments in between, the configuration file looks like this:
	
	``
	devices device p1
	 config
	  cisco-ios-xr:interface GigabitEthernet 0/0/0/0
	   no shutdown
	  exit
	 !
	!
	``
	The file can then be used with the command **load merge** *FILENAME* to achieve the desired results.

move *<path>* [**first** | **last** | **before** *<key>* | **after** *<key>*]	Moves an existing element to a new position in an ordered list. The element can be moved first, last (the default), before, or after another element.
rename *<instance path>* *<new id>*	Renames an instance.

revert	Copies the running configuration to the current configuration and removes all uncommitted changes.
rload(merge|override) (terminal| *<file>***)**	Loads the file relative to the current submode. For example, if a file has a device config, you can enter one device and issue the **rload merge**/**override** *<file>* command to load the config for that device, then enter another device and load the same config file using **rload**. See also the **load** command. • merge—Merges the content of the file or terminal with the current configuration. • override—Replaces the current configuration with the configuration from the file or terminal.
rollback configuration [*<number>*] [*<path>*]	Returns the configuration to a previously committed configuration. You can configure the number of old configurations to store in the wae.conf file. If the configurations to store exceed the threshold, the oldest configuration is removed before creating a new one. The configuration changes are stored in rollback files, where the most recent changes are stored in the file rollbackN with the highest number N. Only the deltas are stored in the rollback files. When rolling back the configuration to rollback N, all changes stored in rollback10001-rollbackN are applied. The optional path argument allows subtrees to be rolled back while the rest of the configuration tree remains unchanged. This command is available only if rollback has been enabled in wae.conf. Example: `user@wae(config)# rollback configuration 10005`
rollback selective [*<number>*] [*<path>*]	Instead of undoing all changes from rollback10001 to rollbackN, you can undo only the changes stored in a specific rollback file. In some cases applying the rollback file might fail, or the configuration might require additional changes in order to be valid. The optional path argument allows subtrees to be rolled back while the rest of the configuration tree remains unchanged.
show full-configuration [*<pathfilter>* [**sort-by** *<idx>*]]	Shows the current configuration, taking local changes into account. The show command can be limited to a part of the configuration by providing a path filter. The **sort-by** argument can be given when the path filter points to a list element with secondary indexes. The name of a secondary index is *idx*. When used, the table is sorted in the order defined by the secondary index, which lets you control the order in which to display instances.
show configuration [*<pathfilter>*]	Shows current edits to the configuration.
show configuration merge [*<pathfilter>* [**sort-by** *<idx>*]]	Shows the current configuration, taking local changes into account. The show command can be limited to a part of the configuration by providing a path filter. The **sort-by** argument can be given when the path filter points to a list element with secondary indexes. The name of a secondary index is *idx*. When used, the table is sorted in the order defined by the secondary index, which lets you control the order in which to display instances.
show configuration commit changes [*<number>* [*<path>*]]	Displays edits associated with a commit, identified by the rollback number created for the commit. The changes are displayed as forward changes, as opposed to **show configuration rollback changes**, which displays the commands for undoing the changes. The optional path argument allows only edits related to a given subtree to be listed.
show configuration commit list [*<path>*]	Lists rollback files. The optional path argument allows only rollback files related to a given subtree to be listed.

`show configuration rollback listed` [*<number>*]	Displays the operations required to undo the changes performed in a commit associated with a rollback file. These are the changes that are applied if the configuration is rolled back to that rollback number.
`show configuration running` [*<pathfilter>*]	Displays the running configuration without taking uncommitted changes into account. An optional path filter can be provided to limit what is displayed.
`show configuration diff` [*<pathfilter>*]	Displays uncommitted changes to the running configuration in diff-style, with + and - in front of added and deleted configuration lines.
`show parser dump` *<command prefix>*	Shows all possible commands that start with the command prefix.
`tag add` *<statement>* *<tag>*	Adds a tag to a configuration statement. This command is available only when the system is configured with attributes enabled.
`tag del` *<statement>* *<tag>*	Removes a tag from a configuration statement. This command is available only when the system is configured with attributes enabled.
`tag clear` *<statement>*	Removes all tags from a configuration statement. This command is available only when the system is configured with attributes enabled.

`timecmd` *<command>*	Measures and displays the execution time of a command. This command is available only if devtools has been set to true in the CLI session settings. Example:
	
	`user@wae# timecmd id`
	`user = admin(501), gid=20, groups=admin, gids=12,20,33,61,79,80,81,98,100`
	`Command executed in 0.00 sec`
	`user@wae#`

`top` [*<command>*]	Exits to the top level of configuration, or executes a command at the top level of the configuration.
`unhide` *<hide-group>*	Unhides all elements and actions that belong to the hide-group. A password might be required. This command is hidden and is not shown during command completion.
`validate`	Validates the current configuration. This is the same operation as **commit check**.
`xpath` [`ctx` *<path>*] (`eval` | `must` | `when`) *<expression>*	Evaluates an XPath expression. A context-path can be used as the current context for the evaluation of the expression. If no context-path is given, the current sub-mode is used as the context-path. The pipe command trace can be used to display debug or trace information. This command is available only if devtools has been set to true in the CLI session settings. • eval—Evaluates an XPath expression. • must—Evaluates the expression as a YANG *must* expression. • when—Evaluate the expression as a YANG *when* expression.

Expert Mode and WAE CLI Comparison

Although this guide describes many configurations using the Expert Mode, it is important to note that you can use the Expert Mode and CLI interfaces interchangeably. The advantage of using the Expert Mode, other

than the GUI, is that it displays all available fields for configuration. In the CLI, you must know the parameters or view the CLI command help to see all available options.

Configuration in the CLI follows the same path structure as navigating in the Expert Mode. The following table lists some equivalent CLI commands and Expert Mode configuration with sample data.

Configuration Type	Expert Mode	CLI Equivalent

Create a device authgroup with device credentials.	1. From Expert Mode, **Configuration editor**, navigate to **/ncs:devices** and click the **authgroups** tab.
	
	2. Click **group**.
	
	3. Click the plus (+) sign, enter `groupABC` as the authgroup name, and click **Add**.
	
	4. Click **default-map** and enter following authentication parameters: **remote-name**—`rpc1`, **remote-password**—`XLydrf`, **remote-secondary-password**—`XLydrr`.
		`# devices authgroups group `**`groupABC`**
		` default-map`
		`remote-name `**`rpc1`**` remote-password`
		`XLydrf`
		`remote-secondary-password `**`XLydrr`**

Create a network model by discovering the network using XTC (topo-bgpls-xtc-nimo). **Note** This example assumes that the network access and an XTC agent have been configured and are running.	1. From Expert Mode, **Configuration editor**, navigate to **/wae:networks**, click the plus (+) sign, and enter `as54001_topo`. 2. Click **Add**. 3. Click the **nimo** tab and choose **topo-bgpls-xtc-nimo** as the NIMO type. 4. Enter the following: | Field	User Input
---	---
network-access	`as54001`
xtc-host	`xt11`
backup-xtc-host	`xt12`
asn	`54001`
igp-protocol	`isis`

extended-topology-discovery	`true`		`# networks network `**`as54001_topo`**` nimo`
			` topo-bgpls-xtc-nimo`
			`network-access `**`as54001`**` xtc-host`
			`xtc11`
			`backup-xtc-host `**`xtc12`**` igp-protocol`
			`isis`
			`extended-topology-discovery `**`true`**` asn`
			` 54001`

Configuration Type	Expert Mode	CLI Equivalent
Consolidate network models.	1. From Expert Mode, **Configuration editor**, navigate to **/wae:networks**, click the plus (+) sign, and enter and **as54001**. 2. Click **Add**. 3. Click the **nimo** tab and choose **aggregator**. 4. Click **aggregator** > plus (+) sign, and choose the source NIMOs: **as54001_topo**, **as54001_xtclsp**, **as54001_conflsp**, and **as54001_snmplsp**.	`# networks network `**`as54001`**` nimo aggregator sources `**`as54001_topo`** `# networks network `**`as54001`**` nimo aggregator sources `**`as54001_xtclsp`** `# networks network `**`as54001`**` nimo aggregator sources `**`as54001_conflsp`** `# networks network `**`as54001`**` nimo aggregator sources `**`as54001_snmplsp`**

Configure a Network Model Using the WAE CLI

This workflow describes the high-level configuration steps on how to create a network model using the Expert Mode.

Step	For more information, see...
1. Configure device authgroups and SNMP groups.	Configure Device Access Using the CLI, on page 48
2. Configure a network access profile.	Configure a Network Access Profile, on page 50
3. Configure agents. **Note** This step is only required for collecting XTC or multi-layer information.	• Configuring XTC Agents Using the Expert Mode, on page 28 • Configure the Configuration Parsing Agent Using the Expert Mode, on page 32
4. Create a network and collect basic topology data.	Create a Network Model, on page 51
5. Configure additional data collections or NIMO capabilities.	Configure Additional NIMOs, on page 52
6. (Optional) Configure the Scheduler.	Scheduler Configuration, on page 149
7. Configure and view plan archives.	Configure the Archive Using the WAE CLI, on page 52

Configure Device Access Using the CLI

WAE uses `authgroups` for login and SNMP access to devices. The following procedure describes how to configure device access using the CLI in configuration mode.

Procedure

Step 1   Create device authgroup(s) with device credentials.

```
# devices authgroups group <group_name>
# default-map remote-name <username>
# default-map remote-password <user_password>
# default-map remote-secondary-password <secondary_password>
# commit
```

Step 2   Create SNMP group(s) to be able to run SNMP tools.

```
# devices authgroups snmp-group <group_name>
# default-map community-name <community_name>
# commit
```

For SNMPv3, you can set the following options:

```
# devices authgroups snmp-group <group_name>
# default-map community-name <community_name>
# default-map usm remote-name <remote_user>
# default-map usm security-level <auth-priv or auth-no-priv or no-auth-no-priv>
# default-map usm auth <auth_protocol> remote-password <remote_password>
# default-map usm priv <priv_protocol> remote-password <remote_password>
# commit
```

Note   Even though there are options to select authentication and encryption with *no-auth-no-priv*, these values
are not used in the backend and are optional.

Example

For example (using simple names and passwords for demonstration purposes):

```
user@wae(config)# devices authgroups group ABCgroup default-map remote-name anyuser
remote-password password123 remote-secondary-password mypassword
user@wae(config)# commit

user@wae(config)# devices authgroups snmp-group snmp_v2 default-map community-name mycompany
user@wae(config)# commit

user@wae(config)# devices authgroups snmp-group snmp_v3_01
user@wae(config)# default-map community-name mycompany
user@wae(config)# default-map usm remote-name User1
user@wae(config)# default-map usm security-level auth-priv
user@wae(config)# default-map usm auth md5 remote-password pass_a123
user@wae(config)# default-map usm priv aes remote-password pass_a123
user@wae(config)# commit

user@wae(config)# devices authgroups snmp-group snmp_v3_02
user@wae(config)# default-map community-name mycompany
user@wae(config)# default-map usm remote-name User2
user@wae(config)# default-map usm security-level auth-no-priv
user@wae(config)# default-map usm auth sha remote-password pass_b456
user@wae(config)# commit

user@wae(config)# devices authgroups snmp-group snmp_v3_03
```

```
user@wae(config)# default-map community-name mycompany
user@wae(config)# default-map usm remote-name User2
user@wae(config)# default-map usm security-level no-auth-no-priv
user@wae(config)# commit
```

Configure a Network Access Profile

Before you begin

Confirm that authentication and SNMP groups have been configured. For more information, see Configure Device Access Using the CLI, on page 48.

Procedure

Step 1 Enter the following commands:

```
# wae nimos network-access network-access <network-access-ID> auth-group <auth-group-ID>
# wae nimos network-access network-access <network-access-ID> snmp-group <snmp-group-ID>
```

Step 2 Repeat the following command to enter each management IP address:

```
# wae nimos network-access network-access <network-access-IP> node-access <node-access-ID-1> auth-group
 <auth_group_ID> default-snmp-group <snmp-group-ID>
ip-manage <ip-address-1>
```

Note Node filter does not work with ip-manage.

Step 3 Commit the configuration:

```
# commit
```

Example

For example:

```
# wae nimos network-access network-access as64001 auth-group ABCgroup
# wae nimos network-access network-access as64001 snmp-group snmp_v3_01
# wae nimos network-access network-access as64001 node-access 10.1.1.1 ip-manage 10.18.20.121
# wae nimos network-access network-access as64001 node-access 10.2.2.2 ip-manage 10.18.20.122
# commit

# wae nimos network-access network-access netaccess_01 auth-group ABCgroup
# wae nimos network-access network-access netaccess_01 snmp-group snmp_v2
node-access 122.168.200.2 ip-manage 192.18.20.2
```

Create a Network Model

When creating a network you must also configure basic topology collection using the topo-igp-nimo or the topo-bgpls-xtc-nimo. For more information, see Topology Collection Using the IGP Database, on page 58 and Topology Collection Using XTC, on page 59.

![note icon]

Note   You have the option to load an existing plan file to create a network model. See Load Plan Files, on page 52.

Before you begin

- Device access and network access must be configured.

- If creating a network running XTC, confirm that XTC agents have been configured. For more information, see Configuring XTC Agents Using the Expert Mode, on page 28.

Procedure

Enter the following commands:

```
# networks network <topo-network-model-name> nimo <NIMO-name>
network-access <network-access> <parameter-1>
<parameter-1-option> <parameter-2> <parameter-2-option>
<parameter-x> <parameter-x-option>
# commit
```

Example

The following examples show two ways to configure the topo-bgpls-xtc-nimo.

Example 1:

```
# networks network NetworkABC_topo-bgpls-xtc-nimo nimo topo-bgpls-xtc-nimo network-access
TTE_lab_access
# networks network NetworkABC_topo-bgpls-xtc-nimo nimo topo-bgpls-xtc-nimo xtc-host TTE-xtc11
# networks network NetworkABC_topo-bgpls-xtc-nimo nimo topo-bgpls-xtc-nimo backup-xtc-host
 xtc12
# networks network NetworkABC_topo-bgpls-xtc-nimo nimo topo-bgpls-xtc-nimo asn 62001
# networks network NetworkABC_topo-bgpls-xtc-nimo nimo topo-bgpls-xtc-nimo igp-protocol
isis
# networks network NetworkABC_topo-bgpls-xtc-nimo nimo topo-bgpls-xtc-nimo
extended-topology-discovery true
```

Example 2:

```
# networks network NetworkABC_topo-bgpls-xtc-nimo nimo topo-bgpls-xtc-nimo
network-access TTE_lab xtc-host TTE-xtc11
backup-xtc-host TTE-xtc12 igp-protocol isis
extended-topology-discovery true asn 62001
```

What to do next

Configure additional network collections using this network model as the source network. For more information, see NIMO Descriptions, on page 55.

Load Plan Files

You can load plan files to create a network model. This is useful if, for example, you already have a plan file with collected topology and demand information. Instead of starting from scratch by creating a new network model with basic topology collection and then augmenting it with demands, you can load an existing plan file.

Use the following command to create a network model from a plan file:

```
# wae components load-plan run plan-file <plan-file-location> network-name
<network-model-name>
```

For example:

```
# wae components load-plan run plan-file /home/tommy/us_atlanta_wan1.txt network-name
NetworkABC_topo_demands
```

Configure Additional NIMOs

This topic describes the general steps to configure different types of advanced network data collection. NIMOs are used to collect different types of data. Some NIMOs require the configuration of agents. For more information, see NIMO Descriptions, on page 55.

Before you begin

You must have a network model with basic collection to be used as a source network.

Procedure

Enter the following command:

```
# networks network <network-model-name> nimo <NIMO-name> source-network <source-network>
```

```
# networks network <network-model-name> nimo <NIMO-name> <parameter-x> <parameter-x-option>
```

Configure the Archive Using the WAE CLI

You can also Configure the Archive Using the Network Model Composer, on page 22.

Procedure

Step 1   Launch the WAE CLI and enter configuration mode.

```
# wae_cli -C
# config
(config)#
```

Step 2 Configure the archive directory and select whether to get archive data from a file.

```
(config)# networks network <network_model_name> plan-archive archive-dir <archive_directory>
(config)# networks network <network_model_name> plan-archive source <file>
(config)# commit
```

For example:

```
(config)# networks network Network_123 plan-archive archive-dir /archive/planfiles/Network_123
(config)# networks network <network_model_name> plan-archive source filename
(config)# commit
```

Step 3 Run archive. This saves the current network model in a plan file (.pln format) into the archive directory you specified.

```
(config)# networks network <network_model_name> plan-archive run
```

For example:

```
(config)# networks network Network_123 plan-archive run
status true
message Successfully archived plan file 20170131_1919_UTC.pln for network Network_123
```

Step 4 Confirm that the plan file was saved by going to the archive directory. The archive directory is divided into the following subfolders: years, months, and days.

For example:

```
(config)# ls /Network_123/2017/01/31
20170131_0100_UTC.pln 20170131_0330_UTC.pln 20170131_1012_UTC.pln
20170131_1312_UTC.pln 20170131_1919_UTC.pln
```

What to do next

Schedule how often a plan file is saved to the Archive.

Manage Plan Files in Archive

Confirm that the archive has been configured and an archive directory has been created. For more information, see .

You can perform the following tasks in archive:

- To list all plan files:

  (config)# networks network <network_model_name> plan-archive list

- To save the network model to archive:

  (config)# networks network <network_model_name> plan-archive run

- To retrieve a plan file:

```
(config)# networks network <network_model_name> plan-archive get
```

Note   You can use an existing plan file to create a network model. See Load Plan Files, on page 52.

Network Interface Modules (NIMOs)

The following sections describe how to configure different types of network collection and other NIMO capabilities. Although the following topics show how to use the Expert Mode for configuration, you can also use the WAE UI or WAE CLI. The topics describe the options that can be configured using any interface.

NIMO Descriptions

Each NIMO has capabilities (derived from NETCONF protocol capabilities) that determine what it collects or deploys. The following table lists a description of each NIMO.

To list the capabilities of each NIMO, click the **get-capabilities** button (in the Expert Mode) after a NIMO is configured.

Note If you wish to consolidate different data collections (NIMO collections) under a single network model, configure the aggregator before running any collections. For more information, see NIMO Collection Consolidation, on page 62.

Collection or Capability	NIMO	Description	Prerequisite/Notes
Network Collection NIMOs			
Topology Collection Using the IGP Database, on page 58	topo-igp-nimo	Discovers IGP topology using login and SNMP.	This is a basic topology collection (topology NIMO). The resulting network model is used as the source network for other NIMOs.
Topology Collection Using XTC, on page 59	topo-bgpls-xtc-nimo	Discovers Layer 3 topology using BGP-LS via XTC. It uses raw XTC data as the source for the topology. Node and interface/port properties are discovered using SNMP.	• XTC agents must be configured before running this collection. See Configuring XTC Agents Using the Expert Mode, on page 28. • This is a basic topology collection for networks using XTC. The resulting network model is used as the source network for other NIMOs.
VPN Collection, on page 66	topo-vpn-nimo	Discovers Layer 2 and Layer 3 VPN topology.	A network model with basic topology collection must exist.
BGP Peer Collection, on page 70	topo-bgp-nimo	Discovers BGP peering using login and SNMP.	A network model with basic topology collection must exist.
Port, LSP, SRLG, and VPN Collection Using Configuration Parsing, on page 69	cfg-parse-nimo	Discovers and parses information from router configurations in the network.	• A network model with basic topology collection must exist. • A Configuration Parsing agent must be configured before running this collection. See Configure the Configuration Parsing Agent Using the Expert Mode, on page 32.
Multilayer (L3-L1) Collection, on page 95	optical-nimo	In conjunction with other NIMOs, the final network collection discovers Layer 1 (optical) and Layer 3 topology.	There are configurations that must take place before configuring the optical-nimo. See Expert Mode—Multilayer Collection, on page 96 .
LSP Configuration Collection, on page 66	lsp-config-nimo	Discovers LSPs using NEDs and the LSP binding SIDs via NETCONF.	A network model with basic topology collection must exist.
LSP Collection Using SNMP, on page 72	lsp-snmp-nimo	Discovers LSPs using SNMP.	A network model with basic topology collection must exist.
LSP Collection Using XTC, on page 68	lsp-pcep-xtc-nimo	Discovers PCEP LSPs using XTC.	The Topology Collection Using XTC, on page 59 must be completed before running this collection.

Collection or Capability	NIMO	Description	Prerequisite/Notes
Segment Routing Traffic Matrix Collection, on page 65	sr-traffic-matrix-nimo	Discovers SR LSP traffic information.	• A network model with basic topology collection must exist. • Telemetry must be set up on the router.
NIMO Collection Consolidation, on page 62	—	Aggregates various NIMO information into a single consolidated network model.	Configured network models with information you want to merge into one final network model.
Merge AS Plan Files, on page 88	inter-as-nimo	Plan files from different Autonomous Systems (AS) can be merged using the inter-as-nimo. The inter-as-nimo resolves any conflicts across the plan files. Plan files in native format are also supported.	• Confirm that collection has been completed on the individual AS network models that you want to merge. • Any AS network models that use the topo-bgpls-xtc NIMO must each have an Autonomous System Number (ASN) assigned to it.
Additional NIMOs			
Traffic Collection, on page 80	traffic-poll-nimo	Collects traffic statistics (interface measurements) using SNMP polling.	• A network model with basic topology collection. • If collecting LSP traffic, a network model with LSP collection must exist. See LSP Collection Using SNMP, on page 72. • If collecting VPN traffic, a network model with VPN collection must exist. See VPN Collection, on page 66.
Network Model Layout (Visualization), on page 84	layout-nimo	Adds layout properties to a source model to improve visualization.	• A consolidated network model. • After the layout-nimo is configured, a plan file containing layout properties must be imported back into the layout-nimo model.
Running External Scripts Against a Network Model, on page 89	external-executable-nimo	Runs customized scripts to append additional data to a source network model.	A source network model and a custom script.

Basic Topology Collection

The network model resulting from basic topology collections (topology NIMOs) is used as the source network for additional data collections. To consolidate topology and other data collections, you must first set up the aggregator before running any collection. For more information on the aggregator, see NIMO Collection Consolidation, on page 62.

Topology Collection Using the IGP Database

The IGP topology (topo-igp-nimo) discovers network topology using the IGP database with the collection of node properties and interface and port discovery using SNMP. This is typically the first NIMO that is configured before other NIMOs, because it provides the basic data collection needed. This NIMO provides full topology discovery. Collection of multi instances of OSPF and ISIS is also supported. All links collected from routers will have an associated IGP process ID.

The network model resulting from this topology discovery is used as the source network for additional collections. It provides the core node, circuit, and interface information used by other NIMOs.

Note
- It is assumed that you are in the middle of creating a network model when performing the tasks described in this topic. For more information, see Create a Network Model, on page 33.

- Although this topic shows how to use the Expert Mode for configuration, it can be referred to for configuring options using the WAE UI or WAE CLI.

Before you begin

Device and network access profiles must be configured. See Configure Network Access, on page 28.

Procedure

Step 1 Choose **topo-igp-nimo** as the NIMO type.

Step 2 Choose a network access profile.

Step 3 From the collect-interfaces field, choose **true** to discover the full network topology.

Step 4 Click the **igp-config** tab to configure seed routers.

Step 5 Click the plus (+) sign to add an IGP.

Step 6 Click **Add** and enter an index number.

   a) Enter the management IP address of the seed router.
   b) Select the IGP protocol that is running on the network.
   c) (Optional) Click the **advanced** tab to configure additional parameters for that IGP configuration. Hover the mouse pointer over fields to view descriptions.

Step 7 (Optional) To add more IGP configurations, navigate back to the igp-config tab and repeat the previous step for each IGP index.

Step 8   (Optional) To exclude or include individual nodes from collection, click the **node-filter** tab and select the node filter. If you have not defined the node-filter, see Configure the Node Filter using Cisco WAE UI, on page 15

Step 9   Expert users can click the advanced tab for more options. Hover the mouse pointer over fields to view option descriptions.

Step 10   Click the **Commit** button.

Step 11   Click **run-collection** > **Invoke run-collection**.

Step 12   To verify that the collection ran successfully, navigate to back to the network (**/wae:networks/network/<*network-name*>**) and click the **model** tab.

Step 13   Click **nodes**. A list of nodes and details appears, indicating a successful collection.

What to do next

Use this network model as the source network to configure additional collections. See NIMO Descriptions, on page 55.

Topology Collection Using XTC

Table 1: Feature History

Feature	Release Information	Description
Enhanced Model Updates Based on SR-PCE Notifications	Cisco WAE Release 7.5.0	The topo-bgpls-xtc-nimo is now enhanced to capture network updates for any changes in IGP-metric, Delay and Node Overload.

The topo-bgpls-xtc-nimo discovers Layer 3 topology using BGP-LS via XTC. It uses raw XTC data as the source for topology. Node and interface/port properties are discovered using SNMP. For testing purposes, you can also use BGP-LS XTC topology discovery using XTC only (extended topology discovery disabled) when no SNMP access is available. The network model resulting from topology discovery is used as the source network for additional collections because it provides the core node/circuit/interface information used by other NIMOs.

The topo-bgpls-xtc-nimo also captures network updates for any changes in IGP-metric, Delay and Node Overload.

BGP-LS XTC topology discovery *using XTC only* is used as a source for only some NIMOs because it does not collect the necessary information needed by most NIMOs.

Before you begin

- Device access and network access must be configured. For more information, see Configure Device Access Using the Expert Mode, on page 27 and Configure Network Access, on page 28.

- An XTC agent must be configured and running. For more information, see Configuring XTC Agents Using the Expert Mode, on page 28.

Procedure

Step 1   From the Expert Mode, in **Configuration editor**, navigate to **/wae:networks**.

Step 2   Click the plus (+) sign and enter a network model name. We recommend a unique name that contains the NIMO name; for example, networkABC_bgpls_xtc.

Step 3   Click **Add**.

Step 4   Click the **nimo** tab.

Step 5   From the **Choice - nimo-type** drop-down list, choose **topo-bgpls-xtc-nimo**.

Step 6   Enter the following information:

> - **network-access**—Choose the network access.
> - **xtc-host**—Choose an XTC agent.
> - **backup-xtc-host**—Choose a backup XTC agent. You can enter the same XTC agent if you do not have a backup.
> - **asn**—Enter 0 to collect information from all autonomous systems in the network, or enter the autonomous system number (ASN) to collect information only from a particular ASN. For example, if the XTC agent has visibility to ASN 64010 and ASN 64020, enter 64020 to collect information only from ASN 64020. You must enter an ASN if you plan to use the as-merger NIMO to consolidate different AS models into one network model.
> - **igp-protocol**—Choose the IGP protocol that is running on the network.
> - **extended-topology-discovery**—Choose **true** to discover the full network topology (node and interfaces).

> **Note**   For more information on advanced options, see BGP-LS XTC Advanced Options, on page 61. From the WAE UI, you can also hover your mouse over each field to view tooltips.

Step 7   Click **reactive-network** tab to subscribe to notifications from XTC to update Node/Link addition or deletion. Enter the following information:

> - **enable**—Select true to enable notifications from XTC to update Node/Link addition or deletion.
> - **enable-triggering-collection**—Select true to collect topology collection on new topology additions.
> - **trigger-debounce-time**—Set the time to wait for last trigger notification before triggering topology collection.

Step 8   Click the **Commit** button.

Step 9   Click **run-xtc-collection** > **Invoke run-collection**.

Step 10 To verify that the collection ran successfully, navigate to back to the network (**/wae:networks/network/<*network-name*>**) and click the **model** tab.

Step 11 Click **nodes**. A list of nodes and details appears, indicating a successful collection.

Example

For example, if using the WAE CLI (in config mode), enter:

```
# networks network <network-model-name> nimo topo-bgpls-xtc-nimo network-access
<network-access-ID>
# networks network <network-model-name> nimo topo-bgpls-xtc-nimo xtc-host <XTC-agent>
# networks network <network-model-name> nimo topo-bgpls-xtc-nimo backup-xtc-host
<XTC-agent-backup>
# networks network <network-model-name> nimo topo-bgpls-xtc-nimo asn <ASN-number>
# networks network <network-model-name> nimo topo-bgpls-xtc-nimo igp-protocol
```

```
<IGP-protocol-type>
# networks network <network-model-name> nimo topo-bgpls-xtc-nimo extended-topology-discovery
 <true-or-false>
```

What to do next

After performing this task, you can use this network model as the source network to configure additional collections. For more information, see NIMO Descriptions, on page 55.

BGP-LS XTC Advanced Options

This topic describes advanced options available when running BGP-LS topology collection using XTC.

Option	Description
nodes	
remove-node-suffix	Remove node suffixes from node names if the node contains this suffix. For example, 'company.net' removes the domain name for the network.
nodes | interfaces	
net-recorder	If set to 'record', SNMP messages to and from the live network are recorded in the net-record-file as discovery runs. Used for debugging.
net-record-file	Directory in which to save the SNMP record. Used for debugging.
interfaces	
find-parallel-links	Find parallel links that aren't in the IGP database (when IS-IS TE extensions aren't enabled).
ip-guessing	Level of IP address guessing to perform for interfaces that are not present in the topology database. (Used when IS-IS TE extensions aren't enabled.) • off—Perform no guessing. • safe—Choose guesses that have no ambiguity. • full—Make best-guess decisions when there is ambiguity.
lag	Enable LAG discovery of port members.
lag-port-match	Determine how to match local and remote ports in port circuits. • exact—Match based on LACP. • none—Do not create port circuits. • guess—Create port circuits to match as many ports as possible. • complete—Match based on LACP first, and then try to match as many as possible.
cleanup-circuits	Remove circuits that don't have IP addresses associated to interfaces. Circuit removal is sometimes required with IS-IS databases to fix IS-IS advertising inconsistencies.

Option	Description
copy-descriptions	Copy physical interface descriptions to logical interfaces if there is only one logical interface and its description is blank.
get-physical-ports	Collect L3 physical ports for Cisco. Collect physical ports if there is an L1 connection underneath.
min-prefix-length	Minimum prefix length to allow when finding parallel links. All interfaces with equal or larger prefix lengths (but less than 32) are considered.
min-guess-prefix-length	Minimum IP guessing prefix length. All interfaces with equal or larger prefix lengths are considered.

NIMO Collection Consolidation

The aggregator uses the Delta Aggregation Rules Engine (DARE) to combine user-specified NIMOs into a single consolidated network model. The aggregator reads the capabilities of source NIMOs. For more information on aggregator functions, see Network Models, on page 3.

Note

- For networks using XTC, you can get automated network updates to obtain real-time network models that can be used for automation applications. For more information, see Automation Applications, on page 141.

- Adding multiple networks of same nimo type under an aggregator is not supported. If the external executable nimo is configured, nimo type must specified under **aggregator/sources/source/nimo**.

- **resync-aggregator-net** does a fresh aggregatioiin of all the NIMO and populates the DARE and SAGE network afresh. **reset-final-network** only populates the SAGE network afresh.

Before you begin

- Configure NIMOs that you want to include in the final network model.

- It is important not to run a collection or execute these NIMOs until after the initial configuration. If collection is run before configuring DARE, then the DARE network should be rebuilt from scratch (**/wae:wae/components/aggregators/aggregator** *<network_name>* and click **resync-aggregator-net**.

- Configuration of NIMO aggregation is simplified when using the Network Model Composer. For more information, see Use the Network Model Composer, on page 16 and Consolidate NIMO Collections Using the Network Model Composer, on page 19 topics.

Procedure

Step 1 Create an empty network. This will be the final consolidated network model. From the Expert Mode, in **Configuration editor**, navigate to **/wae:networks**, click the plus (+) sign, and enter a final network model name.

Step 2 Navigate to **/wae:wae/components/aggregators** and select the aggregator tab.

Step 3 Click the plus (+) sign.

Step 4 From the drop-down destination list, select the final network and click **Add**.

Step 5 Click the source link.

Step 6 Click the plus (+) sign to add source NIMOs and enter the following information:

- **nimo**—Enter NIMO type.
- **direct-source**—If set to false, changes to this model will not be aggregated into the final model. By default, this is set to true.
- **filter-capabilities**—Sets whether or not the capability filter is applied before aggregation. If set to false, all changes will be included for aggregation. For example, the external-executable-nimo does not expose capabilities so this field would be set to false.

> **Note** Configure sr-traffic-matrix-nimo for Bandwidth on Demand and Bandwidth Optimization. See Segment Routing Traffic Matrix Collection, on page 65.

Step 7 (Optional) Continue to add all source NIMOs you want to consolidate collections for under the final network model.

Step 8 (Optional) To configure aging parameters, navigate to **/wae:wae/components/aggregators** and click the **aging** tab.

- **aging-enabled**— Select true to enable aging.
- **l3-node-aging-duration**—Enter the time duration for which an L3 node must be kept in the network after it becomes inactive.
- **l3-port-aging-duration**—Enter the time duration for which an L3 port must be kept in the network after it becomes inactive.
- **l3-circuit-aging-duration**—Enter the time duration for which an L3 circuit must be kept in the network after it becomes inactive.

> **Note** The time duration for l3-node-aging-duration must be greater than l3-port-aging-duration which in turn must be greater than l3-circuit-aging-duration.

- **l1-node-aging-duration**—Enter the time duration for which an L1 node must be kept in the network after it becomes inactive.
- **l1-port-aging-duration**—Enter the time duration for which an L1 port must be kept in the network after it becomes inactive.
- **l1-link-aging-duration**—Enter the time duration for which an L1 link must be kept in the network after it becomes inactive.

> **Note** The time duration for l1-node-aging-duration must be greater than l1-port-aging-duration which in turn must be greater than l1-link-aging-duration.

Step 9 Click the **Commit** button.

Step 10 Run the source NIMOs. The final network model will update with the latest information from the source network models. See also Aggregator and Multilayer Collection Configuration Example, on page 64.

Example

If using the WAE CLI (in config mode), enter:

```
# wae components aggregators aggregator <final-network-model>
# sources source <nimo_1>
```

```
# sources source <nimo_2>
# dependencies dependency <demands-network>
# dependencies dependency <inventory-network>
# dependencies dependency <layout-network>
# dependencies dependency <traffic-network>
# final-network <sage-network>
# commit
```

After the aggregator is configured, then run the source NIMOs.

Aggregator and Multilayer Collection Configuration Example

This example shows how to configure the aggregator to combine Layer 3 and Layer 1 network model information using the CLI.

The following shows that L1 (optical) and L3 (topo-igp-nimo) network models have been configured on the network. For more information on how to configure the optical NIMO and topo-igp-nimo, see the following topics: , .

Layer 1 network model:

```
networks network l1-network
    nimo optical-nimo source-network l3-network
    nimo optical-nimo network-access cisco_access
    nimo optical-nimo optical-agents cisco_network
        advanced use-configure-l3-l1-mapping true
        advanced l3-l1-mapping    bg1_mapping
    !
```

Layer 3 network model:

```
networks network l3-network
    nimo topo-igp-nimo network-access bgl-lab-access
    nimo topo-igp-nimo igp-config 1
    seed-router 10.225.120.61
    igp-protocol isis
    !
    nimo topo-igp-nimo collect-interfaces true
    nimo topo-igp-nimo advanced interfaces lag true
    !
```

Note Collection has not yet been done on the configured L1 and L3 network models.

Configure the aggregator.

```
# config
# wae components aggregators aggregator l1-l3-final-model
# sources source l1-network
# sources source l3-network
# dependencies dependency dmd-network
# dependencies dependency inv-network
# dependencies dependency lyt-network
# dependencies dependency traffic-network
```

```
# final-network sage-1
# commit
```

After the aggregator is configured, run the L3 and L1 collections.

```
# networks network l3-network nimo topo-igp-nimo run-collection
```

Run the L1 network collection.

```
# networks network l1-network nimo optical-nimo build-optical-topology
```

Open WAE Design to view the final network model (**File** > **Open from** > **WAE Automation Server** and select the final network model to verify data collection.

Segment Routing Traffic Matrix Collection

Segment Routing (SR) Traffic Collection (sr-traffic-matrix-nimo) discovers SR traffic. This NIMO enables the generation of demands between external interfaces of a network from collected telemetry data.

Before you begin

- A basic topology network model must exist. See Topology Collection Using the IGP Database, on page 58 or Topology Collection Using XTC, on page 59.

- Telemetry must be configured. See the Configure Telemetry in WAE, on page 137 topic.

Procedure

Step 1 From the Expert Mode, in **Configuration editor**, navigate to **/wae:networks**.

Step 2 Click the plus (+) sign and enter a network model name. We recommend a unique name that contains the source network and NIMO names; for example, networkABC_sr_traffic_matrix.

Step 3 Click **Add**.

Step 4 Click the **nimo** tab.

Step 5 From the **Choice - nimo-type** drop-down list, choose **sr-traffic-matrix-nimo**.

Step 6 Click **sr-traffic-matrix-nimo** and enter the source network, collection period.

>   **Note**   The source network must be an aggregated network with all LSP data in it if LSP demands are to be generated.

Step 7 Click the **Commit** button.

Step 8 Click **run-collection** > **Invoke run-collection**.

Example

If using the WAE CLI (in config mode), enter:

```
# networks network <network-model-name> nimo sr-traffic-matrix-nimo source-network
<source-network>

# networks network <network-model-name> nimo sr-traffic-matrix-nimo run-collection
# commit
```

VPN Collection

The VPN Collection (topo-vpn-nimo) discovers Layer 2 and Layer 3 VPN topology.

Before you begin

Network topology collection must be complete. For more information, see .

Procedure

Step 1	From the Expert Mode, in **Configuration editor**, navigate to **/wae:networks**.
Step 2	Click the plus (+) sign and enter a network model name. We recommend a unique name that contains the source network and NIMO names; for example, networkABC_vpn.
Step 3	Click **Add**.
Step 4	Click the **nimo** tab.
Step 5	From the **Choice - nimo-type** drop-down list, choose **topo-vpn-nimo**.
Step 6	Click **topo-vpn-nimo** and enter the following:

- **source-network**—Choose the applicable network model that contains basic topology information.
- **network-access**—Choose the network access.

Step 7	Click the **vpn-types** tab.
Step 8	Click the plus (+) icon to add at least one VPN type:

- **VPWS**—Add this type when Virtual Private Wire Service is being used in the network.
- **L3VPN**—Add this type when Layer 3 VPN is being used in the network.

Step 9	Click the **Commit** button.
Step 10	Navigate back to the **topo-vpn-nimo** tab and click **run-collection** > **Invoke run-collection**.

LSP Configuration Collection

LSP configuration information in the network is collected using the LSP Configuration NIMO (lsp-config-nimo) and the LSP NSO Agent (cisco-wae-nso-agent). The LSP NSO Agent manages interactions between NSO instances, retrieves LSP information, and converts it into a WAE network model representation.

The LSP NSO Agent also keeps a history and diagnostic files of network models in *<wae_run_directory>*/packages/cisco-wae-nso-agent/res/files:

- merged-full.xml – Contains data collected on the most recent network model.

- merged-full-last.xml – Contains data collected on the previous network model.

- patch.xml - Contains only the content differences (delta) from the previous network model to the most recent network model.

- filtered_*<network_name>*.xml – Contains a filtered version of the merged-full.xml using only nodes from the specified network *<network_name>*.

- Temporary diagnostic files from the last NETCONF query:

  - nso_config_address_last.xml

  - nso_config_explicit_path_last.xml

  - nso_config_segment-list.last.xml

  - nso_config_tunnels_last.xml

Before you begin

A basic topology network model must exist. For more information, see .

Procedure

Step 1   From the WAE CLI, configure NSO instances so that the LSP NSO Agent knows where to collect from under wae/agents/nso-agent/nso-servers.

```
admin@wae# config
Entering configuration mode terminal
admin@wae(config)# wae agents nso-agent nso-servers <NSO_instance_name> host <host_ip_address> port
 <netconf_ssh_port> user <user> password <password>
admin@wae(config)# commit
```

The following fields must be configured:

- NSO instance name—This can be any unique string identifier for the NSO instance.

- host—The NSO instance IP address or hostname.

- port—The NETCONF SSH port that the NSO instance uses. This is found in netconf-north-bound/transport/ssh/port/ncs.conf. The default value is 2022.

- user—The NSO user that is authorized to access the NETCONF API. The default user is "admin".

- password—The NSO user password.

Step 2   Collect LSP information from the NSO instances.

```
admin@wae# wae agents nso-agent get-config-netconf
```

Note   Typically, this should be a scheduled task to periodically collect LSP information.

Step 3   From the Expert Mode, in **Configuration editor**, navigate to **/wae:networks**.

Step 4   Click the plus (+) sign and enter a network model name. We recommend a unique name that contains the source network and NIMO names; for example, networkABC_lsp_config.

Step 5   Click **Add**.

Step 6   Click the **nimo** tab.

Step 7   From the **Choice - nimo-type** drop-down list, choose **lsp-config-nimo**.

Step 8   Click **lsp-config-nimo** and enter the source topology network.

> **Note**   The lsp-config-nimo must follow a topology NIMO. For example, you cannot choose a layout network (layout-nimo) as the source network.

Step 9   Click the **Commit** button.

Step 10   Click **run-collection** > **Invoke run-collection**.

LSP Collection Using XTC

LSP discovery using XTC (lsp-pcep-xtc-nimo) uses the data collected from the bgpls-xtc-nimo and appends LSP information, thus creating a new network model.

Before you begin

Confirm that BGP-LS topology collection using XTC (bgpls-xtc-nimo) has been completed for a network. You will need to use this model as the source network for collecting LSPs. For more information, see Topology Collection Using XTC, on page 59.

Procedure

Step 1   From the Expert Mode, in **Configuration editor**, navigate to **/wae:networks**.

Step 2   Click the plus (+) sign and enter a network model name. We recommend a unique name that contains the source network and NIMO names; for example, `networkABC_lsp_pcep_xtc`.

Step 3   Click **Add**.

Step 4   Click the **nimo** tab.

Step 5   From the **Choice - nimo-type** drop-down list, choose **lsp-pcep-xtc-nimo**.

Step 6   Click **lsp-pcep-xtc-nimo** and enter the source network. This is the network model that contains topology information collected using the bgpls-xtc-nimo.

Step 7   Click the **xtc-hosts** tab.

Step 8   Click the plus (+) icon and enter the following:

- **name**—Enter an XTC hostname. This can be any arbitrary name.
- **xtc-host**—From the drop-down list, choose one of the XTC hosts that was previously configured. For more information, see Configuring XTC Agents Using the Expert Mode, on page 28.

Step 9   Click **reactive-network** tab to subscribe to notifications from XTC to update LSPs based on addition or deletion. Enter the following information:

- **enable**—Select true to enable notifications to modify network topology.

- **enable-triggering-index-collection**—Select true to trigger collection of tunnel indexes, through SNMP, on new tunnels.
- **trigger-debounce-time**—Set the time to wait for last trigger notification before triggering tunnel index collection.
- **network-access**—Enter the network access profile for the network.
- **connect-timeout**—Enter timeout in minutes.
- **verbosity**—Enter the log verbosity level.
- **net-recorder**—Select the SNMP record action. Default is off.
- **net-record-file**—Enter SNMP record filename.

Step 10   Click the **Commit** button.

Step 11   Click **run-xtc-collection** > **Invoke run-collection**.

Step 12   To verify that the collection ran successfully, navigate to back to the network (**/wae:networks/network/<*network-name*>**) and click the **model** tab.

Step 13   Click **nodes**. A list of nodes and details appears, indicating a successful collection.

Step 14   Choose one of the nodes that you know has an LSP and click the **lsps** tab.

Step 15   Click the **lsp** link. A table with a list of discovered LSPs appears.

Port, LSP, SRLG, and VPN Collection Using Configuration Parsing

This topic covers the cfg-parse-nimo NIMO.

Note   cfg-parse-nimo NIMO is not a base topology collector. It must only be used to augment details missing from other methods of collection like SNMP, XTC.

Before you begin

- A topology network model must exist. See Create a Network Model, on page 33.

- The Configuration Parsing agent must be configured and running. For more information, see Configure the Configuration Parsing Agent Using the Expert Mode, on page 32.

Procedure

Step 1   From the Expert Mode, in **Configuration editor**, navigate to **/wae:networks**.

Step 2   Click the plus (+) sign and enter a network model name. We recommend a unique name that contains the source network and NIMO names; for example, networkABC_port-cfg-parse.

Step 3   Click **Add**.

Step 4   Click the **nimo** tab and choose cfg-parse-nimo as the NIMO type.

Step 5   Click NIMO link and enter the following information:

- **source-network**—Choose the applicable network model that contains topology information.

- **source**—Choose between cfg-parse-agent or directory. If you have used cfg-parse-agent to get the configs select cfg-parse-agent option, and then choose a configuration parse agent. Else, if you have the configs in a directory, select directory option and enter the directory where the configurations are stored

Step 6   Click the **parse** tab.

Step 7   Enter configuration parse values. To view field descriptions, hover the mouse pointer over the field name. More information on some of the fields are described below:

- **igp-protocol**—Choose which interfaces are part of the topology: IS-IS and/or OSPF-enabled interfaces. The default is ISIS.

- **ospf-area**—The agent can read information for single or multiple areas. The ospf-area option specifies the area ID or all. The default is area 0.

- **isis level**—The agent can read IS-IS Level 1, Level 2, or both Level 1 and Level 2 metrics. If both are selected, the agent combines both levels into a single network. Level 2 metrics take precedence.

- **asn**—ASN is ignored by default. However, for networks that span multiple BGP ASNs, use this option to read information from more than one IGP process ID or instance ID in an ASN.

Click **include-object** to add collection types: lag, srlg, rsvp, vpn, frr, sr_lsps, lmp, and sr_policies.

Note
- `l2vpn` config parse is not supported.

- When `l3vpn` information is collected by cfg-parse NIMO, it is assumed that all VPNs are connected to each other.

- If cfg-parse NIMO is collecting VPN information and topo-vpn NIMO is also being run, make sure that topo-vpn NIMO is before cfg-parse NIMO in the NIMO chain.

- Single ended SRLGs with other end missing will be collected via SR-PCE. SRLGSCircuits table is not updated for the same though.

Step 8   Click the **Commit** button.

Step 9   Click **run-collection** > **Invoke run-collection**.

What to do next

After performing this task, you can use this network model as a source network to configure additional collections. For more information, see NIMO Descriptions, on page 55.

BGP Peer Collection

The topo-bgp-nimo discovers BGP topology via SNMP and login. It uses a topology network (typically an IGP topology collection model) as its source network and adds BGP links to external ASN nodes.

Before you begin

A topology network model must exist. See Create a Network Model, on page 33.

Procedure

Step 1 From the Expert Mode, in **Configuration editor**, navigate to **/wae:networks**.

Step 2 Click the plus (+) sign and enter a network model name. We recommend a unique name that contains the source network and NIMO names; for example, networkABC_topo_bgp.

Step 3 Click **Add**.

Step 4 Click the **nimo** tab.

Step 5 From the **Choice - nimo-type** drop-down list, choose **topo-bgp-nimo**.

Step 6 Click **topo-bgp-nimo** and enter the following information:

- **source-network**—Choose the applicable network model that contains basic topology information.
- **network-access**—Choose a network access profile that was previously configured.
- **min-prefix-length**—(Optional) Enter the min-prefix-length to control how restrictive IPv4 subnet matching is in discovering interfaces as BGP links.
- **min-IPv6-prefix-length**—(Optional) Enter the min-IPv6-prefix-length to control how restrictive IPv6 subnet matching is in discovering interfaces as BGP links.
- **login-multi-hop**—(Optional) Choose whether to disable login-multihop if you do not want to log in to routers that potentially contain multihop peers.

For more information on advanced options, see .

Step 7 Click the **peer-protocol** tab and select the type of peer discovery to be used. Choose between IPv4, IPv6 or both.

Step 8 Click the **Commit** button.

Step 9 Click **run-collection** > **Invoke run-collection**.

BGP Topology Advanced Options

This topic describes advanced options available when running BGP topology collection.

Option	Description
force-login-platform	Override platform detect and use the specified platform. Valid values: cisco, juniper, alu, huawei.
fallback-login-platform	Fallback vendor in case platform detection fails. Valid values: cisco, juniper, alu, huawei.
try-send-enable	When logging in to a router, send an enable password if the platform type is not detected. This action has the same behavior as '-fallback-login-platform cisco'.
internal-asns	Specify internal ASNs. If used, the specified ASNs are set to internal; all others are set to external. The default is to use what is discovered. **Note:** This option is available only in WAE CLI.

Option	Description

asn-include	Specify external ASNs of interest. If specified, BGP peer discovery from the source network is restricted to the listed external ASNs. If no asn-include is configured, topo-bgp-nimo will discover BGP peers with all external ASNs from the current source network.
	
	Note: This option is available only in WAE CLI. You can configure multiple ASNs as comma separated values. For example:
	
	`admin@wae(config)#networks network TOPO-BGP nimo topo-bgp-nimo`
	`advanced asn-include 1111,2222`

find-internal-asn-links	Find links between two or more internal ASNs. Normally this action is not required because IGP discovers these links.
find-non-ip-exit-interface	Search for exit interfaces that are not represented as next-hop IP addresses, but rather as interfaces (which are rare). **Note** This action increases the amount of SNMP requests for BGP discovery, which affects performance.
find-internal-exit-interfaces	Collect exit interfaces to internal ASNs.
get-mac-address	Collect source MAC addresses of BGP peers connected to an Internet Exchange public peering switch. This action is required only for MAC accounting.
use-dns	Whether to use DNS to resolve BGP IP addresses.
force-check-all	Check all routers even if there is no indication of potential multi-hop peers. This action could be slow.
net-recorder	If set to 'record', SNMP messages to and from the live network are recorded in the net-record-file as discovery runs. Used for debugging.
net-record-file	Directory in which to save the SNMP record. Used for debugging.
login-record-mode	Record the discovery process. If set to 'record', messages to and from the live network are recorded in the login-record-dir as the tool runs. Used for debugging.
login-record-dir	Directory in which to save the login record. Used for debugging.

LSP Collection Using SNMP

The lsp-snmp-nimo discovers LSP information using SNMP.

Before you begin

A basic topology network model must exist. See .

![Note pencil icon]

Note	Nokia devices are not enabled for LSP stats collection using SNMP by default. They must be enabled for successful collection. Please consult a Nokia representative to enable this option.

Procedure

Step 1	From the Expert Mode, in **Configuration editor**, navigate to **/wae:networks**.
Step 2	Click the plus (+) sign and enter a network model name. We recommend a unique name that contains the source network and NIMO names; for example, networkABC_lsp_config.
Step 3	Click **Add**.
Step 4	Click the **nimo** tab.
Step 5	From the **Choice - nimo-type** drop-down list, choose **lsp-snmp-nimo**.
Step 6	Click **lsp-snmp-nimo** and enter the following:

> • **source-network**—Choose the applicable network model that contains basic topology information.
> • **network-access**—Choose a network access profile that was previously configured.
> • **get-frr-lsps**—Choose **true** if you want to discover Multiprotocol Label Switching (MPLS) Fast Reroute (FRR) LSP (backup and bypass) information.

Step 7	Click the **Commit** button.
Step 8	Click **run-collection** > **Invoke run-collection**.

Inventory Collection

The Inventory Collection (inventory-nimo) collects hardware inventory information.

Collected Hardware

The `get_inventory` tool creates a series of NetIntHardware* tables that store the collected hardware information based on hardware type. While these tables are not directly usable by WAE Live, four of them are processed by `build_inventory` for use in WAE Live. Each of the following objects are defined by node IP address and SNMP ID.

> • NetIntHardwareChassis—Router chassis objects identified by node IP address and SNMP ID.
>
> • NetIntHardwareContainer—Each entry represents a slot in a router (anything that can have a field replaceable unit (FRU) type device installed into it). Examples include chassis slots, module slots, and port slots.
>
> • NetIntHardwareModule—Hardware devices that can be installed into other hardware devices. Generally, these devices directly support traffic such as linecards, modules, and route processors, and do not fall into one of the other function-specific hardware tables.
>
> • NetIntHardwarePort—Physical ports on the router.

Hardware Hierarchy

The hardware has a parent-child relationship based on where the object resides within the router. The chassis has no parent and is considered the *root object* . Other than the chassis, each object has one parent and can have one or more child objects. Objects with no children are called *leaf objects* , such as ports and empty containers. This hierarchy generally reflects how hardware objects are installed within other objects. For instance, a module representing a linecard might have a parent object that is a container representing a slot.

The parent is identifiable in the NetIntHardware* tables by the ParentTable and ParentId columns. Using these two columns along with the Node (node IP address) column, you can find the parent object for any hardware object.

Example: This NetIntHardwareContainer entry identifies that container 172.23.123.456 has a chassis as a parent. In the NetIntHardwareChassis, there is an SnmpID entry that matches the container's ParentId of 2512347.

NetIntHardwareContainer							
Node	SnmpID	ParentID	Model	Name	NumChildren	ParentTable	SlotNumber
172.23.123.456	2503733	2512347		slot mau 0/0/0/5	0	NetIntHardwareChassis	0

Tracing the hierarchy from each leaf object to its corresponding root object based on the parent-child relationships results in a series of object types that form its hardware hierarchy. It is this trace that the `build_inventory` tool uses to determine how to process the hardware devices. This is also the process you must use if adding an entry to the HWInventoryTemplates table.

Example: Chassis-Container-Module-Module-Container-Port

Tables for Processing Inventory

The `build_inventory` tool constructs the NetIntNodeInventory table by processing the NetIntHardware* tables. The tool requires two configuration files and can additionally use an optional one. If not specified, the files included in the *`<run_directory>`*`/packages/cisco-wae-inventory-nimo/priv/etc/inventory` are used.

- master_inventory_templates.txt (required)—This file contains these tables.

    - HWInventoryTemplates entries categorize the devices in the final NetIntNodeInventory table, as well as prune from inclusion.

    - HWNameFormatRules entries format hardware object names to make them more usable, as well as correct unexpected SNMP results.

- master_exclude_list.txt (required)—Contains the ExcludeHWList table that prevents (blocked lists) hardware objects from being included in the final NetIntNodeInventory table. This can be useful when for excluding hardware that does not forward or carry traffic.

- master_hw_spec.txt (optional)—Contains the HardwareSpec table that can be used to adjust collected data in terms of the number of slots in a specified device when the slots returned by SNMP is inaccurate.

If you modify the template or choose to exclude files, you will want these changes to persist across software upgrades.

Configure Hardware Templates

The `build_inventory -template-file` option calls a file containing both the HWInventoryTemplates and the HWNameFormatRules tables, which by default are in the

<run_directory>/packages/cisco-wae-inventory-nimo/priv/etc/inventory/master_inventory_templates.txt file.

HWInventoryTemplates Table

The HWInventoryTemplates table tells the `build_inventory` tool how to interpret hardware referenced by the NetIntHardware* tables. It enables `build_inventory` to categorize objects into common, vendor-neutral hardware types, such as chassis, linecards, and slots, as well as to remove hardware types that are not of interest.

Inventory hardware is categorized as a chassis, slot, linecard, module slot, module, port slot, port, or transceiver. A container is categorized as either a slot, module slot, or port slot. A module is categorized as either a module or a linecard. All other hardware objects are categorized by their same name. For instance, a chassis is categorized as a chassis. These categorized hardware objects are available through the WAE Live application for use in inventory reports.

The build_inventory tool looks at the following columns of the HWInventoryTemplates table for matches in the NetIntHardware* tables in this order.

- DiscoveredHWHierarchy, Vendor, Model

- DiscoveredHWHierarchy, Vendor, * (where * means all entries in the Model column)

You can further enhance the search using the `-guess-template-if-nomatch true` option. In this instance, if no matches are found using the first two criteria, WAE Collector then looks for matches only for DiscoveredHWHierarchy and Vendor, and does not consider Model.

If a match is found, the subsequent columns after DiscoveredHWHierarchy tell `build_inventory` how to categorize the hardware. These latter columns identify hardware object types: chassis, slot, linecard, module slot, module, port slot, port, or transceiver. Each column entry has the following format.

Type,Identifier,Name

- Type is the discovered hardware type, such as "container."

- Identifier specifies which object (of one or more of the same type) in the hierarchy is referenced (0, 1, ...).

- Name specifies a column heading in the NetIntHardware* table. This is the name that appears in for that object in the NetIntNodeInventory table and thus, in WAE Live inventory reports.

Example: Module,0,Model

(Model is a column heading in the NetIntHardwareModule table)

Multiple name source columns can be specified with a colon.

Example: Container,0,Model:Name

If a hardware category does not exist or is empty, `build_inventory` does not include it in the final NetIntNodeInventory table.

Example

Using the first row of the default master_inventory_templates.txt file, WAE Collector searches the NetIntHardware* tables for ones that have entries that match the Vendor, Model, and DiscoveredHWHierarchy columns, as follows.

Cisco ASR9K Chassis-Container-Module-Port-Container-Module

Thereafter, it categorizes each entry in the hardware hierarchy (DiscoveredHWHierarchy column), and defines its location in the hardware types columns.

The first Module entry is defined as a linecard, it is identified as #0, and the name that appears in the NetIntNodeInventory table is the one appearing in the Model column of the NetIntHardwareModule table. The second module is defined as a transceiver object and is identified as #1. It uses the same name format.

Notice that there are two containers in the hierarchy, but there is only one defined as a Type. This means that the second container would not appear in the NetIntNodeInventory table.

Add HWInventoryTemplates Entries

If WAE Collector encounters an inventory device that is not in the HWInventoryTemplates table, it generates a warning that specifies pieces of the hardware hierarchy, including the SNMP ID of the leaf object and the IP address of the router. You can use this information to manually trace the objects from the leaf to the root and derive an appropriate entry in the HWInventoryTemplates table. For information on tracing hardware hierarchies, see Hardware Hierarchy.

Step 1 Copy the warning message for reference, and use it for Step 2.

Step 2 Using the router's IP address, as well as the SNMP ID, name, and model of the leaf object, find the leaf object referenced in the warning in either the NetIntHardwarePort or the NetIntHardwareContainer table.

Step 3 Use the leaf object's ParentTable and ParentId columns to trace the leaf back to its parent. For each successive parent, use its ParentTable and ParentId columns until you reach the root object (chassis) in the NetIntHardwareChassis table.

Step 4 Once each object in the hardware hierarchy is found, add it to the DiscoveredHWHierarchy column of the HWInventoryTemplates table. Also complete the Vendor and Model columns.

Step 5 For each object in the hardware hierarchy (DiscoveredHWHierarchy column), classify it into one of the standard hardware types, which are the columns listed after the DiscoveredHWHierarchy column.

HWNameFormatRules Table

The HWNameFormatRules table specifies how to format the names in the NetIntNodeInventory table. This is useful for converting long or meaningless names to ones that are easier to read and clearer for users to understand.

For each entry in the HWInventoryTemplates table, the HWNameFormatRules table is searched for a matching vendor, hardware type (HWType), name (PatternMatchExpression). Then, rather than using the name specified in the HWInventoryTemplates table, the NetIntNodeInventory table is updated with the name identified in the ReplacementExpression column.

If multiple matches apply, the first match found is used. Both the PatternMatchExpression and the ReplacementExpression can be defined as a literal string in single quotes or as a regular expression.

Example: The entries in the table work as follows.

- Replaces all Cisco chassis name with 7507 if the name has four characters where A is the beginning of the string and Z is the end of the string.

- Replaces all Cisco linecard names that match 800-20017-.* with 1X10GE-LR-SC.

- Replaces all Juniper chassis named "Juniper (MX960) Internet Backbone Router" with MX960.

HWNameFormatRules			
Vendor	HWType	PatternMatchExpression	ReplacementExpression

Cisco	Chassis	\A4\Z	'7507'
Cisco	Linecard	800-20017-.*	'1X10GE-LR-SC'
Juniper	Chassis	Juniper (MX960) Internet Backbone Router	$1

Note SNMP returns many slot names as text, rather than integers. It is a best practice to remove all text from slot numbers for optimal use in WAE Live inventory reports.

Exclude Hardware by Model or Name

The `build_inventory -exclude-file` option calls a file containing the ExcludeHWList table, which by default is in the
`<run_directory>`/packages/cisco-wae-inventory-nimo/priv/etc/inventory/master_exclude_list.txt
file. This table enables you to identify hardware objects to exclude from the NetIntNodeInventory table based on model, name, or both. This is useful, for instance, when excluding management ports and route processors. The model and names can be specified using regular expressions or they can be literals.

Example: The entries in the table work as follows.

- Exclude all objects in the NetIntHardwarePort table where the vendor is Cisco and the name ends with CPU0/129.

- Exclude all objects in the NetIntHardwareModule table where the vendor is Cisco and the model is 800-12308-02.

- Exclude all objects in the NetIntHardwarePort table where the vendor is Cisco and the name is Mgmt.

ExcludeHWList			
HWTable	Vendor	Model	Name
NetIntHardwarePort	Cisco		\CPU0\129$
NetIntHardwareModule	Cisco	800-12308-02	
NetIntHardwarePort	Cisco		Mgmt

HardwareSpec

The `build_inventory -hardware-spec-file` option calls a file containing the HardwareSpec table, which by default is in the
`<run_directory>`/packages/cisco-wae-inventory-nimo/priv/etc/inventory/master_hw_spec.txt file.
This table enables you to adjust data returned from SNMP. You can adjust both the total number of slots (TotSlot) and the slot numbering range (SlotNum). For instance, SNMP might return 7 slots for a chassis when there are actually 9, including route processors.

This table looks only for hardware that contains slots, module slots, or port slots, and thus, the hardware type (HWType column) must be chassis, linecard, or module. SlotNum indicates the slot number range. For instance, some routers start with slot 0, whereas others start with slot 1.

Example: This table entry sets the Cisco 7609 chassis to have a total of 9 slots and to start the slot numbering with 9.

HardwareSpec				

Vendor	HWType	Model	TotSlot	SlotNum
Cisco	Chassis	7609	9	1-9

Configure Inventory Collection

Before you begin

Network topology collection must be complete. For more information, see .

Procedure

Step 1 From the Expert Mode, in **Configuration editor**, navigate to **/wae:networks**.

Step 2 Click the plus (+) sign and enter a network model name. We recommend a unique name that contains the source network and NIMO names; for example, networkABC_inventory.

Step 3 Click **Add**.

Step 4 Click the **nimo** tab.

Step 5 From the **Choice - nimo-type** drop-down list, choose **inventory-nimo**.

Step 6 Click **inventory-nimo** and enter the following:

- **source-network**—Choose the applicable network model that contains basic topology information.
- **network-access**—Choose the network access.

Step 7 (Optional) Click the **advanced** > **get-inventory-options**. The `get-inventory` tool collects the network hardware and creates NetIntHardware tables that contain every device collected from MIB walks segregated by object type. The tool uses SSH and NETCONF to collect data that is not available in MIBs.

- **login-allowed**—If set to true, allows logging in to the router to collect inventory data.
- **net-recorder**—This option is typically used for debugging. Set to record to record SNMP messages to and from the live network in the net-record-file when discovery is running.
- **net-record-file**—Enter the filename where recorded SNMP messages are saved.

Step 8 (Optional) Click the **advanced** > **build-inventory-options**. The `build-inventory` tool processes the raw hardware data information in the NetIntHardware* tables) to categorize and remove unwanted objects in the final NetIntNodeInventory table.

- **login-allowed**—If set to true, allows logging in to the router using Netconf to collect inventory data. Required only for Juniper transceiver collection.
- **guess-template-if-nomatch**—If set to true, broadens the search when processing raw inventory data.
- **template-file**—Hardware template file containing HWInvetory Templates and HWNameFormatRules tables.
- **hardware-spec-file**—File containing HardwareSpec table that defines slot counts for specific types of hardware to verify SNMP data returned from outers.

Step 9 Click the **Commit** button.

Step 10 Navigate back to the **inventory-nimo** tab and click **run-collection** > **Invoke run-collection**.

Example

WAE CLI (config mode):

```
# networks network <network-model-name> nimo inventory-nimo source-network
<source-network> network-access <network_access_name>
# networks network <network-model-name> nimo inventory-nimo advanced get-inventory-options
 login-allowed <false_or_true>
# networks network <network-model-name> nimo inventory-nimo run-collection
# commit
```

Create auth.enc

The auth.enc has credentials for SNMPv2c, SNMPv3, or both. SNMPv2c uses a less secure security model, passing community strings in clear text. SNMPv3 provides a strong security model that supports authentication, integrity, and confidentiality.

To generate this file, do the following in the WAE CLI (config mode):

```
# wae nimos network-access generate-auth-file network-access <network_access_name>
file-path <directory>/auth.enc
```

where *<directory>* is where you want to save the auth.enc file. For example:

```
# wae nimos network-access generate-auth-file network-access test_lab file-path
/home/wae/auth.enc
message Successfully generated authfile at /home/wae/auth.enc
```

The authorization file password and default seed router login credentials consist of the following.

- primary password—Password for viewing file contents

- login username—Default username for login access to the routers

- login password—Default password for login access to the routers

- login enable password—Default enable password for login access

The SNMPv2c information is defined using a single value.

- community—Default community string

The SNMPv3 information defines authentication and encryption details.

- Security level

    - noAuthNoPriv—Authenticates by username, but does not validate the user and does not encrypt data.

    - authNoPriv—Authenticates by username, validates the user using MD5 or SHA, but does not encrypt data.

    - authPriv—Authenticates by username, validates the user using MD5 or SHA, and encrypts data using DES or AES.

- SNMPv3 username—Username for authentication

- Authentication protocol type—MD5 or SHA

- Authentication password—Password for authentication

- Encryption protocol type—DES or AES

- Encryption password—Password for encryption

- Context name—Name of a collection of management information accessible by an SNMP entity

After you have created the initial encrypted authentication file, you can manually edit the contents to add multiple profiles or communities and map routers to them. Each profile contains a complete set of SNMPv3 authentication and encryption information. Multiple profiles or communities are necessary when different groups of routers use different authentication credentials.

Traffic Collection

The traffic-poll-nimo collects traffic statistics (interface measurements) using SNMP polling.

Before you begin

This NIMO requires the following:

- Basic topology network model.

- If collecting VPN traffic, a VPN network model must exist. See VPN Collection, on page 66.

- If collecting LSP traffic, an LSP network model must exist. See LSP Collection Using SNMP, on page 72.

- Maximum number of open files (ulimit -n): 1,000,000

Limitations

- Node traffic information from external interfaces is not collected.

Procedure

Step 1	From the Expert Mode, in **Configuration editor**, navigate to **/wae:networks**.
Step 2	Click the plus (+) sign and enter a network model name. We recommend a unique name that contains the source network and NIMO names; for example, networkABC_traffic_polling.
Step 3	Click **Add**.
Step 4	Click the **nimo** tab.
Step 5	From the **Choice - nimo-type** drop-down list, choose **traffic-poll-nimo**.
Step 6	Click **traffic-poll-nimo** and enter the following:

- **source-network**—Choose the applicable network model.
- **network-access**—Choose the network access.

Note	If you modify the selected network access after starting the traffic poller, you must restart the traffic poller for the network access changes to take effect.

Step 7 To run continuous traffic collection for interfaces, click the **iface-traffic-poller** tab and enter the following:

- **enabled**—Set to **true**.
- **period**—Enter the polling period, in seconds. We recommend starting with 60 seconds. See Tuning Traffic Polling, on page 82 to tune the polling period.
- **qos-enabled**—Set to **true** if you want to enable queues traffic collection.
- **vpn-enabled**—Set to **true** if you want to enable VPN traffic collection. If set to true, confirm that the source network model has VPNs enabled.

Step 8 To run continuous traffic collection for LSPs, click the **lsp-traffic-poller** tab and enter the following:

- **enabled**—Set to **true**.
- **period**—Enter the polling period, in seconds. We recommend starting with 60 seconds. See Tuning Traffic Polling, on page 82 to tune the polling period.

Step 9 To run continous traffic collection for MAC accounting, click the **mac-traffic-poller** tab and enter the following:

- **enabled**—Set to **true**.
- **period**—Enter the polling period, in seconds. We recommend starting with 60 seconds. See Tuning Traffic Polling, on page 82 to tune the polling period.

Note If **mac-traffic-poller** is enabled, make sure that the source network model has MAC addresses.

Step 10 Click the **Commit** button.

Step 11 Navigate back to the **traffic-poll-nimo** tab and click **run-snmp-traffic-poller** > **Invoke run-snmp-poller**. To stop continuous collection in the future, click **stop-snmp-traffic-poller**.

Traffic Poller Advanced Options

This topic describes advanced options available when configuring traffic collection (traffic-poll-nimo).

Option	Description
snmp-traffic-poller	
stats-computing-minimum-window-length	Enter the minimum window length for traffic calculation, in seconds. The default is 300 seconds.
stats-computing-maximum-window-length	Enter the maximum window length for traffic calculation, in seconds. The default is 450 seconds.
raw-counter-ttl	Enter how long to keep raw counters, in minutes. The default is 15 minutes.
net-recorder	This option is typically used for debugging. Set to **record** to record SNMP messages to and from the live network in the net-record-file when discovery is running.
log-file	Traffic poller log file.
net-record-file	Enter the filename where recorded SNMP messages are saved.

Option	Description
verbosity	Set the poller logging level. The default is 30. • 40—INFO • 50—DEBUG • 60—TRACE
snmp-traffic-population	
scheduler-interval	Enter the interval to perform traffic population, in seconds. The default is 300 seconds. It will send traffic statistics to the configuration database (CDB). If set to 0 (typically set when using the Bandwidth on Demand application), the continuous poller does not calculate and populate traffic automatically. It only calculates and populates the model when `nimo traffic-poll-nimo advanced snmp-traffic-population` is executed. WMD pulls the traffic statistics from RPC API. The traffic statistics are not sent to CDB.
connect-timeout	Enter the maximum execution time for traffic population, in minutes.
kafka-config	
broker-url	URL of Kafka broker.
zookeeper-url	URL of Kafka zookeper.

Tuning Traffic Polling

Traffic poller collects raw traffic counters from the network. Collection time depends on network size, network latency and response time from individual nodes.

To run traffic polling efficiently, do the following:

1. Set the traffic poller logging level to 40.

2. Start with the default options and run continuous collection for several hours. The default values are:

```
iface-traffic-poller/period = 60
lsp-traffic-poller/period = 60
advanced/snmp-traffic-poller/stats-computing-minimum-window-length = 300
advanced/snmp-traffic-poller/stats-computing-maximum-window-length = 450
advanced/snmp-traffic-poller/raw-counter-ttl = 15
advanced/snmp-traffic-population/scheduler-interval = 300
```

3. View the poller.log file. By default, the file is located in `<wae_run_time_directory>/logs/<network_name>-poller.log`.

4. Search for actual collection times. For example:

```
Info [40]: LSP Traffic Poller: Collection complete. Duration: 43.3 sec
Info [40]: Interface Traffic Poller: Collection complete. Duration: 42.7 sec
```

The fastest pace at which the poller can poll network in the example above is around 40-50 secs. This is the minimum value for `iface-traffic-poller->period` and `lsp-traffic-poller->period`. Since

traffic poller populates traffic for both interfaces and lsps at the same time, it is recommended to set both values to the same value.

5. Traffic Poller calculates traffic by collecting raw traffic counters c1, c2, ..., cn. It requires at least two counters to calculate traffic.

```
(c2.counter - c1.counter)/(c2.timestamp - c1.timestamp)
```

6. A sliding window namely `stats-computing-minimum-window-length` is used to sample two counters. It looks for two counters which are farthest apart, that is, latest and earliest for a specified period. The average traffic is calculated for this period. Since the poller requires at least 2 counters, the smallest value for `stats-computing-minimum-window-length` is 2 * `polling period`. To accomodate for variations, add 25% or more.

In case `stats-computing-minimum-window-length` fails to find counters for the specified period due to increased network latency or node response time, it will report traffic as `N/A`. To avoid empty traffic, there is an insurance window, namely `stats-computing-maximum-window-length` which has a minimum value equal to 2 * `polling period`. To accomodate for longer polling period, add 50% or more. For unresponsive nodes add 100% or more.

7. Traffic poller stores raw counters in memory for traffic calculation. This takes up RAM space. Once in a while traffic poller cleans up old counters stored in memory. Anything older than `raw-counter-ttl` (mins) is cleaned up. Therefore given above constraints, minimum value for `raw-counter-ttl` is `stats-computing-maximum-window-length` or more.

8. Traffic population in traffic poller is the process of calculating traffic in the network and populating the plan file/CDB/WMD. The duration it takes depends on network size and target (plan file/CDB/WMD). The fastest target is the plan file (native mode). The actual time it takes to populate traffic can be found in `wae-java-vm` log file. For example:

```
TrafficCalculatorRfs Did-52-Worker-46: - Traffic calculation took (ms) 379976
TrafficCalculatorRfs Did-52-Worker-46: - Traffic calculation took (ms) 391953
TrafficCalculatorRfs Did-52-Worker-46: - Traffic calculation took (ms) 388853
```

In the above example the fastest rate at which traffic can be populated (and consumed by other tools) is about 400 secs.

9. Sometimes in `wae-java-vm` log file you can also see `Invalid counter` warnings to indicate that counter values do not make sense, for example, `c1.counter` is greater than `c2.counter` (which would result in negative traffic). This happens when counters reset or overflow. It is common for 32-bit counters. If there are a lot of them seen, increase the sliding window sizes to process more counters and reduce chances of failure.

10. However it is not recommended to poll network at a faster rate than populating traffic. In the example above the most aggressive setting for traffic polling is 50 secs, but population takes around 400 secs. This amounts to 8 network polls which are wasted. Therefore traffic polling period can be increased (along with sliding window sizes and `raw-counter-ttl`). Here is configuration recommended for the above network:

```
nimo traffic-poll-nimo iface-traffic-poller period 180
nimo traffic-poll-nimo lsp-traffic-poller enabled
nimo traffic-poll-nimo lsp-traffic-poller period 180
nimo traffic-poll-nimo advanced snmp-traffic-poller stats-computing-minimum-window-length
 400
nimo traffic-poll-nimo advanced snmp-traffic-poller stats-computing-maximum-window-length
 800
nimo traffic-poll-nimo advanced snmp-traffic-poller raw-counter-ttl 15
nimo traffic-poll-nimo advanced snmp-traffic-population scheduler-interval 400
nimo traffic-poll-nimo advanced snmp-traffic-population connect-timeout 60
```

> **Note** `snmp-traffic-population connect-timeout` is adjusted to 60 mins for traffic population. This timeout is not used generally and should be just high enough.

Sample configuration above is most aggressive in terms of traffic polling and population. These numbers can be adjusted to be less aggressive to save CPU resources and network bandwidth, for example:

```
nimo traffic-poll-nimo iface-traffic-poller period 240
nimo traffic-poll-nimo lsp-traffic-poller enabled
nimo traffic-poll-nimo lsp-traffic-poller period 240
nimo traffic-poll-nimo advanced snmp-traffic-poller stats-computing-minimum-window-length
 600
nimo traffic-poll-nimo advanced snmp-traffic-poller stats-computing-maximum-window-length
 1200
nimo traffic-poll-nimo advanced snmp-traffic-poller raw-counter-ttl 20
nimo traffic-poll-nimo advanced snmp-traffic-population scheduler-interval 600
nimo traffic-poll-nimo advanced snmp-traffic-population connect-timeout 60
```

Network Model Layout (Visualization)

The layout-nimo adds layout properties to a source network model to improve visualization when importing the plan file into Cisco WAE Design. The NIMO automatically records changes to the layout properties. When the source network model changes, the layout of the destination model is updated.

The layout in the destination network serves as a template that is applied to the source network. The resulting network is saved as the new destination network. If the source layout contains no layout information, the layout from the destination network is simply added to the source network. If the source network contains layout information, that layout is maintained unless there is a conflict with the layout in the destination network. If a conflict exists, the layout information in the destination network takes precedence over the information in the source network.

For example, assume that a new L1 node is added to the source network with a corresponding site assignment. This L1 node is then added to the destination network with its site assignment. Now assume that an existing L1 node has a different site assignment in the source and destination networks. In this case, the site assignment in the destination network is retained.

There are two steps:

1. Create a new network model using the layout-nimo.

2. Add a layout template to the new network model using WAE Design and then send a patch. For more information, see the Cisco WAE Network Visualization Guide.

Before you begin

- Make sure that the Cisco WAE Design version is same or higher than Cisco WAE version on the server.

- A basic topology network model must exist. See .

Procedure

Step 1   From the Expert Mode, in **Configuration editor**, navigate to **/wae:networks**.

Step 2   Click the plus (+) sign and enter a network model name. We recommend a unique name that contains the source network and NIMO names. This procedure uses **networkABC_layout** as an example.

Step 3   Click **Add**.

Step 4   Click the **nimo** tab.

Step 5   From the **Choice - nimo-type** drop-down list, choose **layout-nimo**.

Step 6   Click **layout-nimo** and enter the following:

> • **source-network** - Enter the source network for the network to use.

> • **template-plan-file-path** - Enter the template plan file path from where layout details wiil be copied from.

Step 7   Click the **Commit** button.

Step 8   Launch WAE Design and choose **File** > **Open From** > **WAE Automation Server**.

Step 9   Enter the appropriate details, choose the plan file for the network model you just created (networkABC_layout), and click **OK**.

Step 10   Edit the layout. See the "Using Layouts" chapter in the Cisco WAE Network Visualization Guide.

Step 11   Save the modified plan file to **template-plan-file-path** (see step 6) on the collector server using the option **File** > **Save To** > **WAE Automation Sever** .

Step 12   Click **run-layout** > **Invoke run-layout**.

Step 13   Open a plan from layout network in Cisco WAE Design and confirm if the visual layout is as expected.

Example

WAE CLI (config mode) using the external-executable-nimo:

1. Open the plan file from Cisco WAE design for a network. In this example the source network is NetworkABC_demands.

2. Update the layout.

3. Save the file. In this example, the plan file is named template_01.pln

```
networks network networkABC_layout
nimo layout-nimo source-network NetworkABC_demands
nimo layout-nimo template-plan-file-path /home/centos/plan_files/template_01.pln
nimo layout-nimo advanced advanced-options -method
value visual
!
!
```

Multicast Flow Data Collection

Multicast NIMO collects multicast flow data from a given network. It is a collection of the following NIMOs:

- snmp-find-multicast-nimo—Collects multicast data for multicast flows using SNMP.

- snmp-poll-multicast-nimo—Collects traffic data rate for multicast flows using SNMP.

- login-find-multicast-nimo—Logs in to router to fetch or parse multicast flow data.

- login-poll-multicast-nimo—Logs in to router to get multicast traffic rate

Before you begin

A topology network model must exist. See Create a Network Model, on page 33.

Procedure

Step 1	From the Expert Mode, in **Configuration editor**, navigate to **/wae:networks**.
Step 2	Click the plus (+) sign and enter a network model name. We recommend a unique name that contains the source network and NIMO names; for example, networkABC_multicast.
Step 3	Click **Add**.
Step 4	Click the **nimo** tab.
Step 5	Choose the applicable NIMO as the NIMO type. Choose between snmp-find-multicast-nimo, snmp-poll-multicast-nimo, login-find-multicast-nimo, login-poll-multicast-nimo.
Step 6	Click NIMO link and enter the following information:

- **network-access**—Choose the network access profile for the network.
- **source-network**—Choose the applicable network model that contains topology information.

Step 7	Click **advanced** tab and enter the information. Hover your mouse over the fields to get more details.
Step 8	Click the **Commit** button.
Step 9	Click **run-collection** > **Invoke run-collection**.

Example

If using WAE CLI, use the following commands:

Configure Multicast NIMO as follows with one NIMO as source to next one.

```
networks network snmp_find_mc
nimo snmp-find-multicast-nimo network-access network_access
nimo snmp-find-multicast-nimo source-network dare_network
!
networks network login_find_mc
nimo login-find-multicast-nimo network-access network_access
nimo login-find-multicast-nimo source-network snmp_find_mc
!
networks network snmp_poll_mc
nimo snmp-poll-multicast-nimo network-access network_access
nimo snmp-poll-multicast-nimo source-network login_find_mc
!
networks network login_poll_mc
nimo login-poll-multicast-nimo network-access network_access
nimo login-poll-multicast-nimo source-network snmp_poll_mc
!
```

In aggregator configuration, mark `snmp-find` and `snmp-poll` multicast networks as indirect (direct-source as False).

```
wae components aggregators aggregator  dare_network
sources source mcast-topo
  nimo topo-igp-nimo
!
dependencies dependency login_find_mc
  nimo login-find-multicast-nimo
!
dependencies dependency login_poll_mc
  nimo login-poll-multicast-nimo
!
dependencies dependency snmp_find_mc
  nimo         snmp-find-multicast-nimo
  direct-source false
!
dependencies dependency snmp_poll_mc
  nimo         snmp-poll-multicast-nimo
  direct-source false
!
final-network mcast-final
!
```

Traffic Demands Collection

traffic-demands-nimo collects information regarding traffic demands from the network.

Before you begin

A basic topology network model must exist. See .

Procedure

Step 1   From the Expert Mode, in **Configuration editor**, navigate to **/wae:networks**.

Step 2   Click the plus (+) sign and enter a network model name. We recommend a unique name that contains the source network and NIMO names; for example, networkABC_traffic_demands_config.

Step 3   Click **Add**.

Step 4   Click the **nimo** tab.

Step 5   From the **Choice - nimo-type** drop-down list, choose **traffic-demands-nimo**.

Step 6   Click **traffic-demands-nimo** and enter the following:

- **source-network**—Choose the applicable network model that contains basic topology information.
- **connection-timeout**—Enter connection timeout in mimutes.

Step 7   In the **demand-mesh-config** tab, click **demand-mesh-steps**.

Step 8   Click + to add a step. Enter a name to the step and click **add**.

Step 9   Click the step you just created. Select a tool from **Choice-tool** drop down menu.

Step 10   Click the tool and enter the necessary details.

Step 11   Click **advanced** tab and enter the details. Hover the mouse pointer over fields to view option descriptions.

Repeat steps 9 to 11 to add more steps to the configuration.

Step 12   Click the **Commit** button.

Step 13   Click **run-collection** > **Invoke run-collection**.

Merge AS Plan Files

Plan files from different Autonomous Systems (AS) can be merged using the **inter-as-nimo**. The **inter-as-nimo** resolves any conflicts across the plan files. Plan files in native format are also supported.

Each AS can be on a different WAE server.

Note
- Only Autonomous Systems (AS), Circuits, Nodes, Interfaces, EExternal Endpoints, External Endpoint Members with virtual nodes and unresolved interfaces are

- The following demands are resolved:
  - Source or Destination associated with virtual node that are resolved with real node.
  - Source or Destination associated with the interface in a specific format.
  - Source or Destination associated with the External Endpoints.

- The following demands are not resolved:
  - Source or Destination associated with ASN number only.

- For a given plan file, the internal AS number must match what other plan files see as an external AS number, and all the Autonomous Systems that are going to be merged need to be discovered in all the plan files.

Before you begin
- Collect topology and traffic information for different Autonomous Systems (AS).
- The plan files from different AS have to be present on the same WAE server and the path to the plan files must be mentioned.

Procedure

Step 1   From the Expert Mode, in **Configuration editor**, navigate to **/wae:networks**.

Step 2   Click the plus (+) sign and enter a network model name. We recommend a unique name that contains the source network and NIMO names; for example, networkABC_merge_as_plan.

Step 3   Click **Add**.

Step 4   Click the **nimo** tab.

Step 5   From the **Choice - nimo-type** drop-down list, choose **inter-as-nimo**.

Step 6   Click **inter-as-nimo** and enter the following details:

- **retain-demands**—Select true to merge the demands.
- **tag-name**—Enter a tag name to help identify the updated rows in .pln file. The tag column in the .pln file gets updated with the tag name for rows that are modified.
- **path-to-report-file**—Enter path to a report file where dropped rows after merge are reported.

Step 7   In the **sources** tab, click + and enter the network. Click **Add**.

Step 8   Enter the **plan-file-path**. If this field is left blank, the NIMO looks up for source with the given name.

Step 9   Click **Commit**.

Step 10   Click **merge-inter-as** > **Invoke merge-inter-as**.

To merge AS plan files using CLI, use the following commands:

```
networks network <network-name>
 nimo inter-as-nimo retain-demands true
 nimo inter-as-nimo tag-name <tag-name>
 nimo inter-as-nimo path-to-report-file <report-file-path>
 nimo inter-as-nimo sources <source1>
  plan-file-path <source1-plan-file-path>
 !
 nimo inter-as-nimo sources <source2>
  plan-file-path <source2-plan-file-path>
 !
!
```

Running External Scripts Against a Network Model

The external-executable-nimo lets you run a customized script against a selected network model. You might want to do this when you want specific data from your network that existing Cisco WAE NIMOs do not provide. In this case, you take an existing model created in Cisco WAE and append information from a custom script to create a final network model that contains the data you want.

For more information, see the following topics:

Before you begin

The configuration of this NIMO is also available in the Cisco WAE UI using the Network Model Composer.

Procedure

Step 1   From the Expert Mode, in **Configuration editor**, navigate to **/wae:networks**.

Step 2   Click the plus (+) sign and enter a network model name.

It is recommended to use a name to match the nimo type or use any existing nimo names to match its capability.

Step 3   Click the **nimo** tab.

Step 4   From the **Choice - nimo-type** drop-down list, choose **external-executable-nimo**.

Step 5   Click **external-executable-nimo** and select the source network.

Step 6   Click the **advanced** tab and enter the following:

- **input-file-version**—Enter the plan file version of the source network model, such as 6.3, 6.4, and so on. Default is the current version.

- **input-file-format**—Specify the plan file format of the source network model. The default is .pln.

- **argv**—Enter arguments (in order) that are required for the script to run. Enter $$input for the source network model, $$output for the resulting network model (after the script runs) and $$authfile for auth file if you are running one of WAE CLI tools (network-access must be configured). It is important to note that $$input, $$output, and other argv arguments must be listed in the order that is required by the script. For an example, see Running External Scripts Example, on page 90.

Step 7   From the external-executable-nimo tab, click **run**.

Example

If using the WAE CLI (in config mode), enter:

```
networks network <network-model-name> nimo external-executable-nimo source-network
<source-network>
advanced argv "<command[s]> <arguments>"
admin@wae(config-network-<network-model-name>)# commit
Commit complete.
admin@wae(config-network-<network-model-name>)# exit
admin@wae(config)# exit

admin@wae# networks network <network-model-name> nimo external-executable-nimo run
```

Running External Scripts Example

This example describes how to use the external-executable-nimo with the WAE CLI. The sample python script (ext_exe_eg.py) appends a description to every interface in the network with "My IGP metric is *<value>*." For another example, see the Configure Inventory Collection, on page 78 topic.

Contents of ext_exe_eg.py:

```
import sys
from com.cisco.wae.opm.network import Network

src  = sys.argv[1]
dest = sys.argv[2]

srcNet =  Network(src)

for node in srcNet.model.nodes:
    cnt = 1
    for iface in node.interfaces:
        iface.description = 'My IGP metric is ' + str(iface.igp_metric)
```

```
        cnt = cnt + 1

srcNet.write(dest)
```

In the WAE CLI, enter:

```
admin@wae(config)# networks network net_dest nimo external-executable-nimo source-network
net_src
advanced argv "/usr/bin/python3 /home/user1/srcs/br1/mate/package/linux/run/ext_exe_eg.py
$$input $$output"
admin@wae(config-network-net_dest)# commit
Commit complete.
admin@wae(config-network-net_dest)# exit
admin@wae(config)# exit

admin@wae# networks network net_dest nimo external-executable-nimo run
status true
message Changes successfully applied.
```

Confirm the script succeeded by checking the network plan file in Cisco WAE Design.

Cisco WAE Modeling Daemon (WMD) Configuration

The Cisco WMD provides a real-time network model in memory. DARE receives network changes (from NIMOs) and sends a patch with these changes to Cisco WMD. For more information on how Cisco WMD and DARE works, see the Overview, on page 1 chapter.

✎

Note   Data that is not available/up to date in WMD is listed in `<run-directory>/packages/cisco-wae-modeling-daemon/priv/etc/sql-patch-config.txt` under the headings `IgnoredTables` and `IgnoredColumns`.

To configure DARE and WMD, see the following topics:

- Configure the WAE Modeling Daemon (WMD), on page 93

Configure the WAE Modeling Daemon (WMD)

WMD provides a near real-time representation (model) of the network in memory so that applications can get access to that model. It gets changes from SAgE.

This procedure describes how to configure WMD using the Expert Mode. However, you can also configure WMD using the WAE UI. From Cisco WAE UI, click **WMD Configuration** and use the information below to configure WMD. Click **Save** to save the configuration.

Before you begin

The following information should be on hand or configured:

- Final network model name
- Design RPC

Procedure

Step 1   From the Expert Mode, in **Configuration editor**, navigate to **/wae:wae/components/wmd:wmd** and click **config**.

Step 2   From the network-name drop-down list, select the final network model.

Step 3   Select **Enable** button to enable WMD.

Step 4   Click **rpc-connection** and enter Design RPC values.

Step 5   Click **app-subscriber-connections** and enter host and port information for all automation application connections.

Step 6   Click **dare** and enter the following values.

- **dare-destination**—Select the DARE network.
- **connection-attempts**—Enter the number of times to try to reconnect until the connection is reestablished.
- **connection-retry-delay**—Enter the interval (in seconds) between connection attempts.

Example

WAE CLI (in config mode) example:

```
# wae components wmd config network-name <final_model_name> dare dare-destination
<dare_model_name>
```

Multilayer (L3-L1) Collection

Multilayer (Layer 3 and Layer 1) network collection is an advanced collection configuration. This section describes how to configure collection from a multilayer network.

After this procedure, you should be able to collect and model the following information:

- Spectrum Switched Optical Networks (SSON) circuit information (central frequency ID, spectral width, sson_enabled, and prefer_lower_frequency_ids) that can be viewed and modeled in WAE Design. The L1Link attribute is also associated with central_frequency_excludelist_id and sson_enabled columns.

- L1 diversity collection

- Notification support for the EPN-M agent which updates the network model on any change in the network without running the full collection

- Topologies from DWDM networks that support Generalized Multiprotocol Label Switching (GMPLS) with non-User Network Interface (UNI) circuits

- L1 circuit paths

- L1 topology with and without amplifiers

- L1 diversity circuits. L1 circuits can be configured to be disjoint from other L1 circuits

- Unprotected and restorable paths

- Actual L1 circuit path hops

- Feasibility metrics and limits

- Inactive L1 links

- L1 node and L1 link SRLGs

- Site information

- User properties

- Aging information and last seen date. To configure aging, see Configure Aging, on page 162.

After collection, you can view the model in Cisco WAE Design or from the Expert Mode: wae:networks/network/*<network_name>*/l1-model

This section contains the following topics:

Multilayer Collection Limitations

The following multilayer (L3-L1) collection limitations exist:

• Multilayer collection for Cisco devices is supported only on the following platforms:

  • NCS 2000 platforms running version 10.9, 11.1 and 11.2 are supported when using the Cisco Evolved Programmable Network Manager optical agent (EPN-M optical agent) (Configure Multilayer Collection Using the EPN-M Agent, on page 98).

  • Cisco Aggregation Services Routers (ASR) 9000, Cisco Carrier Routing System (CRS), and Cisco NCS 5500 platforms running IOS-XR for L3 devices.

• Multilayer collection is limited to the collection of unprotected circuits.

• L3-L1 mapping by LMP is supported only if the controller interface name is the same as the actual L3 interface name or of the form "dwdmx/x/x/x" where the "x/x/x/x" subscript matches that of the corresponding L3 interface.

• Lambda mapping is currently supported only for circuit paths but not for path hops.

• Only one optical nimo can be configured per WAE instance. If more than one is configured, it is not guaranteed to work.

Expert Mode—Multilayer Collection

Use the topics below to configure a multilayer collection using the Expert Mode. You can also use the Cisco WAE UI (Cisco WAE UI—Multilayer Collection, on page 101) and use these topics for configuration details. To view field descriptions, hover your mouse pointer over fields in the Expert Mode or Cisco WAE UI.

Step	For more information, see...
1. Review multilayer collection limitations.	Multilayer Collection Limitations, on page 96
2. Obtain and configure L3 - L1 mapping information.	Configure L3-L1 Mapping Information, on page 97
3. Configure and run multilayer collection using the Cisco Evolved Programmable Network Manager (EPNM) optical agent. The EPNM agents supports Cisco NCS 2000 series version 10.9, 11.1, and 11.2 devices in your network. You must have EPNM running on your network to use this agent.	• Configure Multilayer Collection Using the EPN-M Agent, on page 98

Configure L3-L1 Mapping Information

L3-L1 mappings can be collected in one of the following ways:

- If LMP is configured on the network, then you can get L3-L1 information by running the configuration parsing agent if LMP is enabled on the network. See Configure the Configuration Parsing Agent Using the Expert Mode, on page 32. The parse configuration agent should be specified in the optical nimo as follows:

  `networks/<multilayer_network_name>/nimo/optical-nimo/advanced/cfg-parse-agent.`

> **Note**
> - L3-L1 mapping by Link Management Protocol (LMP) is supported only if the controller interface name is same as the actual L3 interface name or in the form of "dwdmx/x/x/x" where the "x/x/x/x" subscript matches that of the corresponding L3 interface.
> - EPNM agent does not require the parse config agent to get the L1 L3 mapping.

- Manually configure L3-L1 mapping:

  - Enter the mapping of L3 nodes and interfaces to L1 nodes and ports.

  - Provide an L3 to L1 circuit. This method will discover all the L3 to L1 mappings after topology collection.

> **Note** You must know the L3 and L1 interfaces and ports, or the circuit names.

The following procedure describes the manual configuration of L3 - L1 mapping using the Expert Mode. The Cisco WAE UI can also be used. For more information, see Cisco WAE UI Overview, on page 9.

Procedure

Step 1   From the Expert Mode, in **Configuration editor**, navigate to **/wae:wae/nimos** and click the **l3-l1-mappings** tab.

Step 2   Click **l3-l1-mappings**.

Step 3   Click the plus (+) sign, enter an arbitrary name for the L3-L1 mapping group, and click **Add**.

Step 4   To provide an L3 - L1 circuit:

  a) Click the **l3-l1-circuit-mapping**tab and click the plus (+) sign.
  b) Enter each L3 and L1 circuit names.
  c) Click **Add**.

Step 5   To manually enter the mapping of L3 - L1 interfaces and ports:

  a) Click the **l3-l1-mapping** tab and click the plus (+) sign to enter each mapping.
  b) Click **Add**.

c) Repeat these steps to enter all L3-L1 mappings.

Example

If using the WAE CLI (in config mode):

```
# wae nimos l3-l1-mappings l3-l1-mappings <mapping_name>
l3-l1-circuit-mapping <l3_circuit> <l1_circuit>

# commit
```

Configure Multilayer Collection Using the EPN-M Agent

The Cisco Evolved Programmable Network Manager (Cisco EPN Manager) agent supports Cisco Network Convergence System (NCS) 2000 platforms release 10.9, 11.1, and 11.2 for L1 devices. You must have Cisco EPN Manager version 2.2.2.1 running on your network to use this agent. The agent also receives notifications from Cisco EPN Manager when the status of links and circuits change and updates the network model accordingly. When using the Expert Mode, changes to the nodes, circuits, and so on can be seen in the agent-model tab from the following path:
wae:wae/agents/optical-agent:optical-agents/optical-agent/*<epnm_agent_name>*

Note The different node types reported by EPNM are defaulted to ROADM node type.

Before you begin

.

- Confirm that you have completed all the preliminary tasks in Expert Mode—Multilayer Collection, on page 96.

- Confirm that the EPN-M server certificate is installed. For more information, see the "Security" chapter ("Install a Certificate for the EPN-M Server" topic) in the Cisco WAE Installation Guide.

- (Optional) Configure feasibility limit margins in the Cisco WAE UI or in the Expert mode (**/wae:wae/components/nimos/feasibility-limit-margins/feasibility-limit-margin**). This configuration adds the specified margin to the collected feasibility-limit for the circuits matching the specified bandwidth. For more information, see the "L1 Circuit Wavelengths" Cisco WAE Design User Guide

- (Optional) Configure central frequency blocked lists in the Cisco WAE UI or in the Expert mode (**/wae:wae/components/nimos/central-frequency-excludelists/central-frequency-excludelists**). This configuration defines the list of frequency IDs that may not serve as central frequency IDs for L1 circuit paths. For more information, see Cisco WAE Design User Guide. For information on advanced options and configuration guidelines, see the following topics:

  - L1 Circuit Wavelength Options, on page 103

  - L1 Circuit Wavelength Guidelines, on page 105

  - L1 Circuit Wavelength Configuration Examples, on page 105

Procedure

Step 1   Configure and run an L3 IGP topology collection network model with the following interface options set to **true**:

- lag
- get-physical-ports

> **Note**   For more information, see Topology Collection Using the IGP Database, on page 58.

Step 2   in **Configuration editor**, navigate to **/wae:wae/agents/optical-agents** and click the **optical-agent** tab.

Step 3   Click Add (+) and enter an agent name.

Step 4   From the agent-type drop-down list, select **Cisco-WAE-Optical-EPNM-Agent**.

Step 5   Click the **cisco-wae-optical-epnm-agent** link.

Step 6   Select the **epnm-server-conf** tab and enter the verified domain of the Cisco EPN Manager server and access group (authgroup that was configured in the NCS devices).

> **Note**   Click the advanced tab to enter the L3-L1 mapping group, data recording options (if net-recorder is set to record, the file will be saved in the directory where net-record-dir is set), connection timeout, and pool-size-per-query (number of parallel queries that can be sent to EPNM per L1 element) settings.

WAE CLI (config mode) example:

```
admin@wae(config)# wae agents optical-agents optical-agent
<agent-name> cisco-wae-optical-epnm-agent epnm-server-conf epnm-server-fqdn <fqdn>
epnm-server-access <authgroup>
cisco-wae-optical-epnm-agent advanced net-recorder <net-recorder-option>
cisco-wae-optical-epnm-agent advanced net-record-dir <net-recorder-storage-directory>
cisco-wae-optical-epnm-agent advanced pool-size-per-query <number-of-queries>
cisco-wae-optical-epnm-agent advanced notification subscribe-to-notifications <true or false>
```

Step 7   Click the **Commit** button.

Step 8   If you plan to utilize Lambda ID mapping (where you can set whether channel ID, central frequency, or wavelength will be mapped to the lambda ID), then you must load the Lambda ID configuration file. Enter the following command:

```
# ncs_load -lmj /wae/agents/optical-agents/optical-agent <agent-name>/lambda-mappings
```

Step 9   To run the L1 collection, navigate back to the **cisco-wae-optical-epnm-agent** tab and click **run-collection** > **Invoke run-collection**.

Step 10   Create an L1 optical collection network model:

a) Navigate to **/wae:networks**.

b) Click the plus (+) sign and enter an optical network model name. We recommend a unique name that contains the source network and NIMO names; for example, networkABC_optical.

c) Click **Add**.

d) Click the **nimo** tab.

e) From the **Choice - nimo-type** drop-down list, choose **optical-nimo**.

f) Click **optical-nimo** and enter the following information:

- **source-network**—Choose the applicable network model that contains L3 topology information collected using one of the topology NIMOs.
- **network-access**—Choose a network access group that was previously configured.

Step 11   Click the **advanced** tab to configure the following:

- **cfg-parse-agent**—Choose the configuration parse agent name if it was used for L1-L3 mapping.
- **lag**—Choose true if using the configuration parse agent.
- **feasibility-limit-margin-list**—(Optional) Choose the feasibility margin list name.
- **central-frequency-excludelists**—(Optional) Choose the frequency exclude list.

Step 12 Click the **optical-agents** tab and add any agents that were created.

a) Click the **advanced** tab to configure advanced features, including the following:

- **retain-amplifiers**—Choose true if you want to include amplifiers as part of the collection.
- **map-lambdas**—If set to true, a user table is created that displays the lambda mapping values (LambdaID, ITU channel number, G.694.1, central frequency, and wavelength) selected in the **map-lambda-id-to** field. Cisco recommends to set this value to true when collecting information from a network with L1 links supporting 96 channels.
- **use-configured-l3-l1-mapping**—Choose true if you manually configured the l3-l1 mapping (see Configure L3-L1 Mapping Information, on page 97).
- **l3-l1-mapping**—Choose the l3-l1 mapping group that you configured earlier.
- **collect-user-properties**—Set to false if you do not want to collect user properties from the agent. The default value is true.

Step 13 Configure the aggregator to consolidate the L1 and L3 network models you just created. See example for aggregator rules to pick the data from proper source network. To view a CLI configuration example of the rest of this procedure, see Aggregator and Multilayer Collection Configuration Example, on page 64.

a) Create an empty network. This will be the final consolidated network model. From the Expert Mode, navigate to **/wae:networks**, click the plus (+) sign, and enter a final network model name. For example, networkABC_L3L1.

b) Navigate to **/wae:wae/components/dare:aggregators/aggregator** tab.

c) Click the plus (+) sign and select the multilayer network (networkABC_L3L1) you just created from the destination drop-down list.

d) From the sources tab, click **source**, and add the L1 and L3 network models you want to combine the collections from.

e) Click **Commit**.

Step 14 Run the L3 collection.

a) Navigate to **/wae:networks/network/<*network-name*> nimo/topo-igp-nimo**.

b) From the topo-igp-nimo tab, click **run-collection**.

Step 15 Run the L1 collection.

a) Navigate to **/wae:networks/network/<*network-name*> nimo/optical-nimo**.

b) From the optical-nimo tab, select the L1 source network and click **build-optical-topology**.

Step 16 To verify that the merge was successful, you can open the network from WAE Design (**File** > **Open from** > **WAE Automation Server** and select the final network model).

What to do next

If archive is configured, you can also open the plan file and view L1 and L3 topology using WAE Design. You can then run the optimization tool, make changes to the network, create a patch, etc. For more information on WAE Design, see the Cisco WAE Design User Guide. For more information on archive and plan files, see the following topics:

- Configure the Archive Using the Network Model Composer, on page 22

Cisco WAE UI—Multilayer Collection

The following workflow describes the high-level steps to configure multilayer collection when using the Cisco WAE UI. For more details on configuration options, see the Expert Mode multilayer topics or hover the mouse pointer over fields.

Before you begin

Review multilayer collection limitations.

Procedure

Step 1 Unless VTXP is enabled on the network, configure L3-L1 mapping by doing one of the following:

- For LMP networks, create and configure a configuration parse agent (**Cisco WAE UI** > **Agent Configuration** and select **cfg-parse-agent**).
- Manually enter L3-L1 mapping information (**Cisco WAE UI** > **L3-L1 Mapping**).

Step 2 Configure an optical agent (**Cisco WAE UI** > **Agent Configuration** and select **cisco-wae-optical-epnm-agent**).

Step 3 (Optional) Configure L1 feasibility margins (**Cisco WAE UI** > **Feasibility Limit Margins**).

Step 4 (Optional) Configure central frequency blocked lists (**Cisco WAE UI** > **Central Frequency Excludelists**).

Step 5 Create an L3 topology collection using the IGP database (**Cisco WAE UI** > **Composer** > **Topology** and select **Topo IGP**).

Step 6 Create an L1 topology collection (**Cisco WAE UI** > **Composer** > **Topology** and select **Optical**).

Step 7 Aggregate topology collections (**Cisco WAE UI** > **Composer** > **Aggregation**, confirm L1 and L3 collections are set as "Direct", and click **Rebuild Network**.

Step 8 To verify that the merge was successful, you can open the network from WAE Design (**File** > **Open from** > **WAE Automation Server** and select the final network model).

What to do next

If archive is configured, you can also open the plan file and view L1 and L3 topology using WAE Design. You can then run the optimization tool, make changes to the network, create a patch, etc. For more information on WAE Design, see the Cisco WAE Design User Guide. For more information on archive and plan files, see the following topics:

Cisco WAE CLI—Multilayer Collection

This example shows how to configure multilayer collection using the Cisco WAE CLI (in config mode). This example includes configuration of an EPNM optical agent and manual L1-L3 mapping.

Configure EPNM optical agent

```
wae@wae(config-optical-agent-<optical_agent_name>)#cisco-wae-optical-epnm-agent
epnm-server-conf epnm-server-fqdn <FQDN> epnm-server-access <epnm_auth_group>
```

Create networks

```
wae@wae(config)# networks network <l3_network_name>
wae@wae(config-network-<l3_network_name>)# nimo topo-igp-nimo network-access <access_group>
wae@wae(config-network-<l3_network_name>)# nimo topo-igp-nimo igp-config 1
igp-protocol <ospf/isis> seed-router <seed-ip>
wae@wae(config-network-<l3_network_name>)# commit
wae@wae(config-network-<l3_network_name>)# exit

wae@wae(config)# networks network <ml_network_name>
wae@wae(config-network-<ml_network_name>)# nimo optical-nimo network-access <access_group>
wae@wae(config-network-<ml_network_name>)# nimo optical-nimo source-network <l3_network_name>
wae@wae(config-network-<ml_network_name>)# nimo optical-nimo optical-agents
<optical_agent_name>
wae@wae(config-network-<ml_network_name>)# commit
wae@wae(config-network-<ml_network_name>)# exit

wae@wae(config)# networks network <aggregator_network_name>
wae@wae(config-network-<aggregator_network_name>)# commit
```

Configure L3 topology

```
wae@wae(config)# networks network <network_name>
wae@wae(config-network-<network_name>)# nimo topo-igp-nimo collect-interfaces true
wae@wae(config-network-<network_name>)# nimo topo-igp-nimo advanced interfaces lag true
get-physical-ports true
wae@wae(config-network-<network_name>)# nimo topo-igp-nimo advanced igp isis-level both
login-record-mode playback login-record-dir /home/wae/records/
wae@wae(config-network-<network_name>)# commit
```

Configure L3-L1 mapping (repeat for each mapping)

```
wae@wae(config)# wae nimos l3-l1-mappings l3-l1-mappings <mapping_name> l3-l1-mapping
<l3_node>
<l3_interface> <l1_node> <l3_interface>
wae@wae(config)# networks network <ml_network> optical-nimo optical-agents [<agent_name>]
advanced
use-configured-l3-l1-mapping true l3-l1-mapping <mapping_name>
wae@wae(config-networks-<ml_network_name>)# commit
```

Note: Use the topo-igp-nimo network as source network for optical-nimo.

(Optional) Configure Lambda mapping

Do one of the following:

- Load from a file:

  After the optical agent is created, load (ncs_load or netconf-console --edit-config..) the lambda mapping configuration XML file (lambda-id to channel-id/wavelength/central-frequency mapping) to /wae/agents/optical-agents/optical-agent[agent-name]/lambda-mappings.

  ```
  ncs_load -lmj <file_name>

  netconf-console --edit-config <file_name>
  ```

> **Note** The netconf-console command requires the 'python-paramiko' system package.

- Enable Lambda mapping in the optical nimo and select the base for lambda mapping.

```
wae@wae(config)# networks network <ml_network_name> nimo optical-nimo optical-agents
<optical_agent_name> map-lambdas true map-lambda-id-to
<channel-id/wavelength/central-frequency>
wae@wae(config)# commit
```

(Optional) Configure feasibility limit margin

```
admin@wae(config)# networks network <ml_network_name> nimo optical-nimo optical-agents
<optical_agent_name>
admin@wae(config-optical-agents-<optical_agent_name>)# advanced feasibility-limit-margin-list
 <L1_circuit_bandwidth>
feasibility-limit-margin <margin_value>
```

Repeat the second command to configure multiple margin values for different bandwidths.

Configure DARE (aggregator)

```
admin@wae(config)# wae components aggregators aggregator <aggregator_network_name>
admin@wae(config-aggregator-<aggregator_network_name>)# sources source <l3_network_name>
admin@wae(config-source-<l3_network_name>)# exit
admin@wae(config-aggregator-<l3_network_name>)# sources source <ml_network_name>
admin@wae(config-source-<ml_network_name>)# commit
```

Run L3 topology collection

```
wae@wae# networks network <l3_network_name> nimo topo-igp-nimo run-collection
```

Run L1 topology collection

Start optical plug-in:

```
wae@wae# wae agents optical-agents optical-agent <optical_agent_name>
cisco-wae-optical-epnm-agent run-collection
```

> **Note** Aggregation runs in the background as network collections process.

Generate a plan file

```
wae@wae# wae components getplan run network <network_name> | exclude planfile-content |
save <path/for/plan/file.txt>
```

L1 Circuit Wavelength Options

The following table describes advanced options available for central frequency.

Table 2: L1 Circuit Wavelength Options

Field	Description
The following options are available in **/wae:networks/network /<*network-name*>/nimo/optical-nimo:optical-nimo/advanced/network-options**.	
anchor-frequency	Anchor frequency in THz. Default is 193.1 THz
central-frequency-granularity	Central frequency granularity in GHz. Default is 25 GHz.
central-frequency-excludelists-name	List of names given to set blocked list central frequency IDs mentioned in the central-frequency-id-excludelist table.
frequency-id-lower-bound	The lower bound of the frequency ID.
frequency-id-upper-bound	The upper bound of the frequency ID.
use-pre-configured-excludelist-per-link-type	Use the pre-defined set of frequency blocked list IDs based on the L1 link types: 80 channel, 96 channel, 80 + 96 channel, and Nyquist. The default is true.
The following options are available in **/wae:wae/nimos /optical-nimo:central-frequency-excludelists**.	
name	Name of the central frequency ID blocked list.
type	The channel type associated with the excludelist-80-channel, 96-channel, Nyquist or Other.
frequency-id-lower-bound	The lower bound of the frequency ID associated with the channel type.
frequency-id-upper-bound	The upper bound of the frequency ID associated with the channel type.
central-frequency-excludelist-ids	List of blocked list central frequency IDs.

Frequency Lower and Upper Bounds and Blocked list Values

- An L1 Link supporting 80 channels should **effectively** have:

    - Bounds=[-47, 113]

    - Blocked list IDs={-47, -45, -43, … , 113} (odd IDs)

- An L1 Link supporting 96 channels should **effectively** have:

    - Bounds=[-71, 121]

    - Blocked list IDs={-71, -69, … , 3, … , 121} (odd IDs)

- An L1 Link supporting 96 channels should **effectively** have:

    - Bounds=[-71, 121]

• Blocked list IDs={-71, -69, … , 3, … , 121} (odd IDs)

• A Nyquist Link should **effectively** have:

• Bounds=[-71, 121]

• No Blocked list IDs

L1 Circuit Wavelength Guidelines

The following list provides information that you may find useful when configuring L1 frequency options.

1. The anchor frequency and central frequency granularity will be constant for the given agent network.

2. You can configure the global anchor frequency, central frequency granularity, upper and lower bounds, and blocked list options for each network. Default values should be used only for the anchor frequency, central frequency granularity, and global upper and lower bounds.

3. Different pre-configured blocked lists are provided and corresponds to different L1 Link types (80-channel-system; 96-channel-system; Nyquist-96-channel-system).

Note   Blocked lists include upper and lower bounds.

4. You can select whether or not pre-configured blocked lists are automatically associated to L1 Links. This can be specified in the **use-pre-configured-excludelist-per-link-type**configuration option. By default, this option is set to true.

5. To edit the pre-configured blocked lists, create a new blocked list entry using the **central-frequency-excludelists** configuration option and set the boolean **use-pre-configured-excludelist-per-link-type** to false.

6. Run the **build-optical-topology** action to incorporate the changes done to the network options and blocked list frequency IDs into the plan file.

L1 Circuit Wavelength Configuration Examples

The following are some L1 frequency configuration examples:

1. Do not configure anything specific to the network options or central frequency blocked list IDs. Use the default values.

```
(config)#  networks network <network-name> nimo optical-nimo
optical-agents <agent-name>
```

2. Specify the custom central frequency blocked list name for all the link types used in the network.

```
(config)# networks network <network-name> nimo optical-nimo
advanced network-options use-preconfigured-excludelist-per-link-type false
central-frequency-excludelists-name [ 80-excludelist 96-excludelist nyquist ]
```

```
wae nimos central-frequency-excludelists central-frequency-excludelist 80-excludelist

    channel-type 80-channel-system
    id-list 1,2,3,4,5,6,7,8,9,10
    !
wae nimos central-frequency-excludelists central-frequency-excludelist 96-excludelist
    channel-type 96-channel-system
    id-list 9,11,13,45,80
!
wae nimos central-frequency-excludelists central-frequency-excludelist nyquist
    channel-type nyquist-channel-system
    id-list 5,19,76
!
```

3. Configure different anchor frequency and central frequency granularity for the network.

```
(config)# networks network <network-name> nimo optical-nimo
advanced network-options anchor-frequency <anchor-frequency-value>
central-frequency-granularity <central-frequency-granularity-value>
```

4. Specify the default central frequency, but not the blocked list at the link level.

```
(config)# wae nimos central-frequency-excludelists central-frequency-excludelist my-other

    channel-type other
    id-list 5,19,76
!networks network <nimo-name> nimo optical- nimo advanced network-options
use-preconfigured-excludelist-per-link-type false central-frequency-excludelists-name [
 my-other ]
```

NetFlow Data Collection

This section contains the following topics:

NetFlow Data Collection

WAE can collect and aggregate exported NetFlow and related flow measurements. These measurements can be used to construct accurate demand traffic data for WAE Design. Flow collection provides an alternative to the estimation of demand traffic from interfaces, LSPs, and other statistics using Demand Deduction. NetFlow gathers information about the traffic flow and helps to build traffic and demand matrix. Importing flow measurements is particularly useful when there is full or nearly full flow coverage of a network's edge routers. Additionally, it is beneficial when accuracy of individual demands between external autonomous systems (ASes) is of interest.

Network data collected separately by NIMOs, including topology, BGP neighbors, and interface statistics, is combined with the flow measurements to scale flows and provide a complete demand mesh between both external autonomous systems and internal nodes.

WAE gathers the following types of data to build a network model with flows and their traffic measurements aggregated over time:

- Flow traffic using NetFlow, JFlow, CFlowd, IPFIX, and Netstream flows

- Interface traffic and BGP peers over SNMP

- BGP path attributes over peering sessions

NetFlow Collection Architectures

The Distributed NetFlow (DNF) collection architecture is typically used for large networks. This architecture consists of a JMS broker, controller node, and one or more agents. The broker, and controller node run on the same machine in which WAE collection server is installed. Each agent runs on a different machine. DNF

collection requires a seperate installation of the components like broker, controller node and agents using ansible

The Centralized NetFlow (CNF) collection is typically used for small networks. CNF collection is implemented using DNF collection architecture and consist of JMS broker, controller node, and a single agent runnning on the same machine in which WAE server is installed. The components broker, controller node and agent come preinstalled with the WAE installation server. There is no need for a seperate installation of the components for CNF.

Distributed Netflow (DNF) Collection

The following figures show the DNF architecture and the DNF workflow. In this architecture, each set of network devices exports flow data to a corresponding collection server. The DNF cluster performs flow computation so that each agent is responsible for the flow computation of its corresponding flow collection server that runs the flow collector. The controller node aggregates this information and passes it back to `flow_collector_ias`.

Figure 5: DNF Architecture
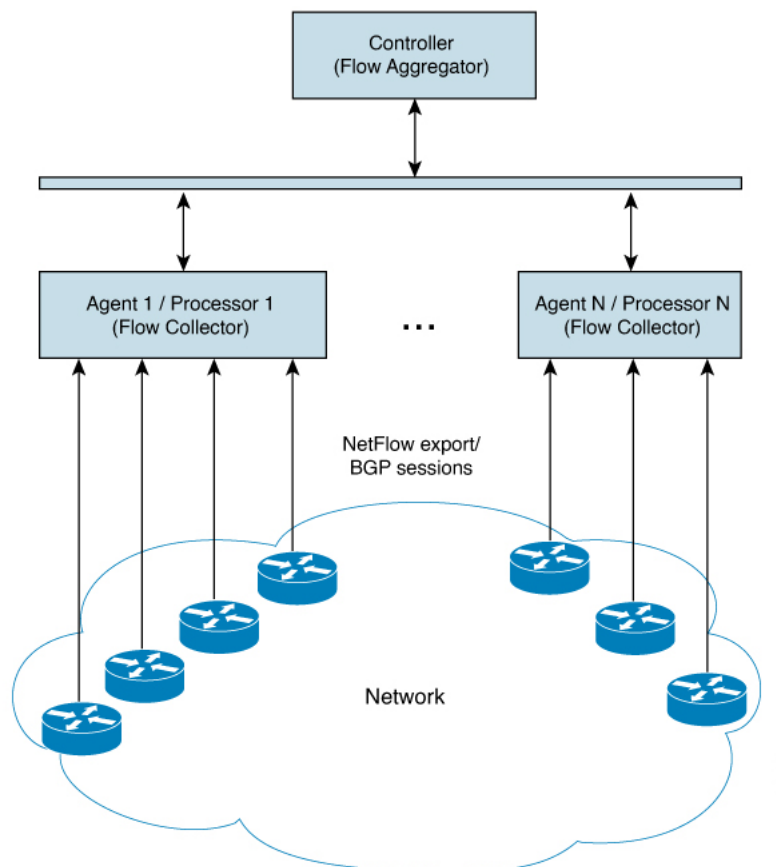
Java Message Server (JMS) Broker

Each distributed flow collection setup has a single JMS broker instance in order for the controller node, agents, and client within a cluster to exchange information. All information is interchanged through the broker and enables all the components to communicate with each other. DNF supports a dedicated JMS broker.

The broker has the following features enabled in order for all JMS clients (controller node, agents, and `flow_collector_ias` instances) to work:

- Out of band file messaging

- Support of obfuscated passwords in configuration files

Controller node and Agents

- Controller node

  The controller node process (`flow_cluster_controller`) provides the following services in the cluster:

  - Monitors and tracks all agents status.

  - Monitors and tracks the status of the last completed IAS computation.

  - Aggregates IAS flow data coming from all agents back to the client (`flow_collector_ias`).

  - Handles configuration and status requests from the cluster (`flow_cluster_manage`).

- Agents

  Only one agent process (`flow_cluster_agent`) per server is supported. Each agent process (`flow_cluster_agent`) receives and computes flow data from its corresponding collection server (`pmacct`).

Figure 6: DNF Collection Workflow
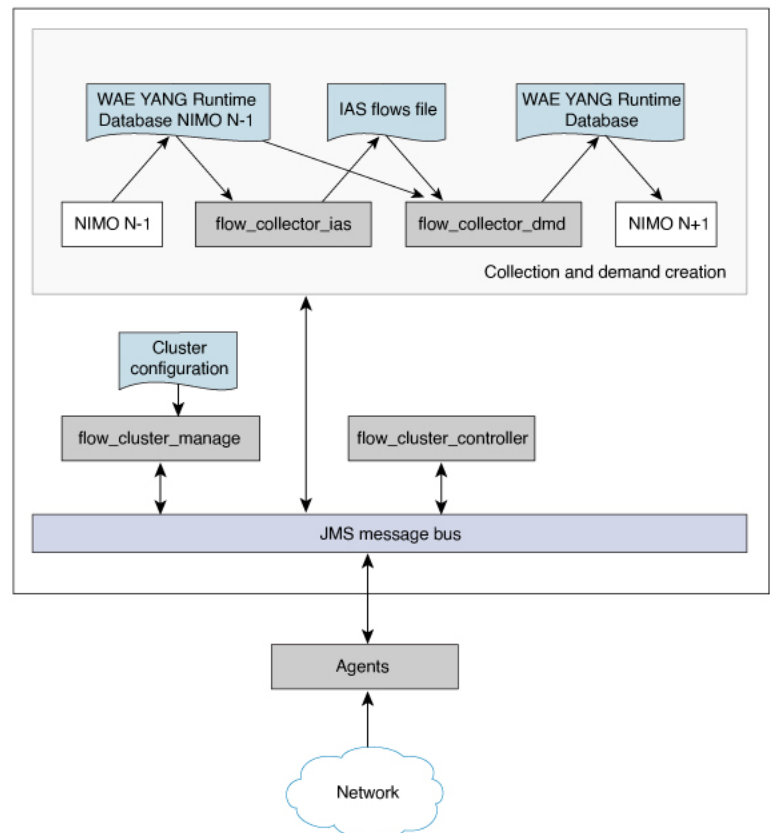
- **flow_cluster_manage**—This CLI tool is used to configure and get status from the cluster. It takes a cluster configuration file and sends the configuration to the cluster.

  A REST API is also available to configure and request status from the cluster as an alternative to using `flow_cluster_manage`. For more information, see the API documentation from one of the following locations:

  - `<wae-installation-directory>`docs/api/netflow/distributed-netflow-rest-api.html

  - http://*<controller-IP-address>*:9090/api-doc For example, to get the cluster configuration:

    For example, to get the cluster configuration:

    ```
    curl -X GET http://localhost:9090/cluster-config > config-file-1
    ```

    For example, to set the cluster configuration:

    ```
    curl -X PUT http://localhost:9090/cluster-config @config-file-2
    ```

    For example, to get the cluster status:

    ```
    curl -X GET http://localhost:9090/cluster-status > config-file-1
    ```

- **flow_cluster_controller**—The controller node service collects all flow data results from all the agents and aggregates the data, which is sent back to `flow_collector_ias`.

- **flow_cluster_agent**—The agent service manages and tracks the status of the associated flow collector. Each agent receives and computes the flow data from its corresponding collection server.

- **flow_cluster_broker**—(not shown in diagram) The JMS broker service allows communication between all components within the architecture, including controller node and agents.

- **flow_collector_ias**—This CLI tool, which is configured inside the `nimo_flow_collector_ias_and_dmd.sh` file and is executed within the external-executable-nimo, receives the flow data from the controller node and produces the IAS flows file.

- **flow_collector_dmd**—This CLI tool sends NetFlow demands and demand traffic to the WAE YANG run-time database. This is configured inside the `nimo_flow_collector_ias_and_dmd.sh` file and is executed within the external-executable-nimo.

Note   In production networks, do not use -log-level=INFO | DEBUG | TRACE for `flow_collector_ias` or `flow_collector_dmd`.

Centralized Netflow (CNF) Collection

The Centralized NetFlow (CNF) collection is typically used for small networks. CNF collection is implemented using DNF collection architecture and consist of JMS broker, controller node, and a single agent runnning on the same machine in which WAE server is installed.

NetFlow Collection Configuration

The flow collection process supports IPv4 and IPv6 flows captured and exported by routers in the ingress direction. It also supports IPv4 and IPv6 iBGP peering.

Routers must be configured to export flows to and establish BGP peering with the flow collection server. Note the following recommendations:

- NetFlow v5, v9, and IPFIX datagram export to the UDP port number of the flow collection server, which has a default setting of 2100. Export of IPv6 flows requires NetFlow v9 or IPFIX.

- Define a BGP session on the routers configured as iBGP Route Reflector Client for the flow collector server. If configuring this in the router itself is not feasible, then a BGP Route Reflector Server with a complete view of all relevant routing tables can be used instead.

- Configure the source IPv4 address of flow export datagrams to be the same as the source IPv4 address of iBGP messages if they are in the same network address space.

- Explicitly configure the BGP router ID.

- If receiving BGP routes, the maximum length of the BGP `AS_path` attribute is limited to three hops. The reason is to prevent excessive server memory consumption, considering that the total length of BGP attributes, including `AS_path`, attached to a single IP prefix can be very large (up to 64 KB).

Centralized NetFlow Configuration Workflow

To configure CNF and start collection:

Note   Unless stated otherwise, do not change permissions on files that were deployed during WAE installation.

Procedure

Step 1   Confirm that the CNF NetFlow Requirements , on page 111 are met.

Step 2   Prepare the Operating System for CNF, on page 112

Step 3   Create the node-flow-configs-table File, on page 112

Step 4   Create the CNF Configuration File, on page 113

Step 5   Configure CNF Collection, on page 116

    a)   Configure the netflow-nimo for CNF, on page 116

CNF NetFlow Requirements

For system requirements, see the *Cisco WAE System Requirements* document.

Licensing

Confirm with your Cisco WAE representative that you have the correct licenses for getting flow and flow demands when using the `flow_cluster_controller`, `flow_collector_ias`, and `flow_collector_dmd` tools.

Prepare the Operating System for CNF

To prepare the OS for CNF, run the following `flow_manage` command from the OS terminal:

```
sudo -E ./flow_cluster_manage -action prepare-os-for-netflow
```

The `prepare-os-for-netflow` option does the following:

- Uses the `setcap` command to allow non-root users limited access to privileged ports (0-1023). This is necessary when configuring the flow collector to use a port under 1024 to listen to BGP messages.

- Configures the OS instance to reserve up to 15,000 of file descriptors to account for the large number of temporary files that may be produced by `flow_collector` tools

✎

Note   After executing this command, you must reboot the server.

Create the node-flow-configs-table File

The <NodeFlowConfigs> table contains basic node configuration information used by the `flow_manage` tool when generating configuration information that it passes to the flow collection server. Thus, prior to executing `flow_manage`, you must construct this table as follows:

- Use a tab or comma delimited format.

- Include one row per node (router) from which you are collecting flow data.

- Enter contents described in the following table for each of these nodes. The BGP columns are required only if collecting BGP information.

Table 3: <NodeFlowConfigs> Table Columns

Column	Description
Name	Node name
SamplingRate	Sampling rate of the packets in exported flows from the node. For example, if the value is 1,024, then one packet out of 1,024 is selected in a deterministic or random manner.
FlowSourceIP	IPv4 source address of flow export packets.
BGPSourceIP	IPv4 or IPv6 source address of iBGP update messages. This column is needed if the `flow_manage -bgp` option is true.
BGPPassword	BGP peering password for MD5 authentication. Use this column if the `flow_manage -bgp` option is true and if BGPSourceIP has a value.

The following is a <NodeFlowConfigs> Table example:

Name	SamplingRate	FlowSourceIP	BGPSourceIP	BGPPassword
paris-er1-fr	1024	192.168.75.10	69.127.75.10	ag5Xh0tGbd7
chicago-cr2-us	1024	192.168.75.15	69.127.75.15	ag5Xh0tGbd7
chicago-cr2-us	1024	192.168.75.15	2001:db9:8:4::2	ag5Xh0tGbd7
tokyo-br1-jp	1024	192.168.75.25	69.127.75.25	ag5Xh0tGbd7
brazilia-er1-bra	1024	192.168.75.30	2001:db8:8:4::2	ag5Xh0tGbd7

Create the CNF Configuration File

To create the cluster configuration file for CNF/DNF, you can use `flow_manage` with `-action produce-cluster-config-file` option, and edit the json file as required:

For example:

Procedure

Step 1 Use the following sample file to create the .json file.

Source the `waerc` file.

```
${CARIDEN_HOME}/flow_manage \
-action produce-cluster-config-file \
-node-flow-configs-table <input-path> \
-cluster-config-file <output-path> \
-interval 120 \
-bgp true \
-bgp-port 10179 \
-port 12100 \
-flow-size lab \
-server-ip ::
```

where `<input-path>` is the path of the `node-flow-configs-table` created under Create the node-flow-configs-table File, and `<output-path>` is the path where you want the resulting seed cluster configuration file to reside.

A sample `<input-path>` file would look like this:

```
<NodeFlowConfigs>
Name    SamplingRate    FlowSourceIP    BGPSourceIP    BGPPassword
node1   1024    192.168.75.10   69.127.75.10    ag5Xh0tGbd7
node2   1024    192.168.75.11   69.127.75.11    ag5Xh0tGbd7
```

Verify that the output of the seed cluster configuration file is similar to the following:

```
{
  "agentConfigMapInfo": {
    "cluster_1::instance_x": {
      "perAgentDebugMode": null,
      "flowManageConfiguration": {
        "maxBgpdPeers": 150,
        "useBgpPeering": true,
        "outfileProductionIntervalInSecs": 120,
```

```
      "networkDeploymentSize": "lab",
      "bgpTcpPort": 10179,
      "netflowUdpPort": 12100,
      "daemonOutputDirPath": "<user.home>/.cariden/etc/net_flow/flow_matrix_interchange",
      "keepDaemonFilesOnStart": false,
      "keepDaemonFilesOnStop": true,
      "purgeOutputFilesToKeep": 3,
      "routerConfigList": [
        {
          "name": "node1",
          "bGPSourceIP": "69.127.75.10",
          "flowSourceIP": "192.168.75.10",
          "bGPPassword": "ag5Xh0tGbd7",
          "samplingRate": "1024"
        },
        {
          "name": "node2",
          "bGPSourceIP": "69.127.75.11",
          "flowSourceIP": "192.168.75.11",
          "bGPPassword": "ag5Xh0tGbd7",
          "samplingRate": "1024"
        }
      ],
      "ipPrefixFilteringList": [],
      "appendedProperties": null,
      "daemonOutputFileMaskPrefix": "out_matrix_",
      "daemonOutputSoftLinkName": "flow_matrix_file-latest",
      "extraAggregation": [],
      "listValidExtraAggregationKeys": false
    }
  },
  "cluster_1::instance_y": {
    "perAgentDebugMode": null,
    "flowManageConfiguration": {
      "maxBgpdPeers": 150,
      "useBgpPeering": true,
      "outfileProductionIntervalInSecs": 120,
      "networkDeploymentSize": "lab",
      "bgpTcpPort": 10179,
      "netflowUdpPort": 12100,
      "daemonOutputDirPath": "<user.home>/.cariden/etc/net_flow/flow_matrix_interchange",
      "keepDaemonFilesOnStart": false,
      "keepDaemonFilesOnStop": true,
      "purgeOutputFilesToKeep": 3,
      "routerConfigList": [
        {
          "name": "node1",
          "bGPSourceIP": "69.127.75.10",
          "flowSourceIP": "192.168.75.10",
          "bGPPassword": "ag5Xh0tGbd7",
          "samplingRate": "1024"
        },
        {
          "name": "node2",
          "bGPSourceIP": "69.127.75.11",
          "flowSourceIP": "192.168.75.11",
          "bGPPassword": "ag5Xh0tGbd7",
          "samplingRate": "1024"
        }
      ],
      "ipPrefixFilteringList": [],
      "appendedProperties": null,
      "daemonOutputFileMaskPrefix": "out_matrix_",
      "daemonOutputSoftLinkName": "flow_matrix_file-latest",
```

```
        "extraAggregation": [],
        "listValidExtraAggregationKeys": false
      }
    }
  },
  "aggregationMode": "okIfNotAllPortionsArePresent",
  "debugMode": {
    "bypassAnyNfacctdOperation": false
  },
  "logNetflowTraffic": false
}
```

Step 2   Edit the file to include only one agent configuration. Below is an example config containing only single agent config.

```
{
  "agentConfigMapInfo": {
    "cluster_1::instance_x": {
      "perAgentDebugMode": null,
      "flowManageConfiguration": {
        "maxBgpdPeers": 150,
        "useBgpPeering": true,
        "outfileProductionIntervalInSecs": 120,
        "networkDeploymentSize": "lab",
        "bgpTcpPort": 10179,
        "netflowUdpPort": 12100,
        "daemonOutputDirPath": "<user.home>/.cariden/etc/net_flow/flow_matrix_interchange",
        "keepDaemonFilesOnStart": false,
        "keepDaemonFilesOnStop": true,
        "purgeOutputFilesToKeep": 3,
        "routerConfigList": [
          {
            "name": "node1",
            "bGPSourceIP": "69.127.75.10",
            "flowSourceIP": "192.168.75.10",
            "bGPPassword": "ag5Xh0tGbd7",
            "samplingRate": "1024"
          },
          {
            "name": "node2",
            "bGPSourceIP": "69.127.75.11",
            "flowSourceIP": "192.168.75.11",
            "bGPPassword": "ag5Xh0tGbd7",
            "samplingRate": "1024"
          }
        ],
        "ipPrefixFilteringList": [],
        "appendedProperties": null,
        "daemonOutputFileMaskPrefix": "out_matrix_",
        "daemonOutputSoftLinkName": "flow_matrix_file-latest",
        "extraAggregation": [],
        "listValidExtraAggregationKeys": false
      }
    }
  },
  "aggregationMode": "okIfNotAllPortionsArePresent",
  "debugMode": {
    "bypassAnyNfacctdOperation": false
  },
  "logNetflowTraffic": false
}
```

Configure CNF Collection

Configure the netflow-nimo for CNF

Before you begin

- You must have a source network model. This is the final network model which includes topology collection and any other NIMO collections you want to include.

- Configure WAE netflow agents to operate in single mode. See

Procedure

Step 1   From the Expert Mode, navigate to **/wae:networks**.

Step 2   Click the plus (+) sign and enter a network model name. We recommend a unique name that is easily identifiable; for example, networkABC_CNF_flow_get.

Step 3   Click the **nimo** tab.

Step 4   From the **Choice - nimo-type** drop-down list, choose **netflow-nimo**.

Step 5   Click **netflow-nimo** and select the **source-network**.

Step 6   Click the **config** tab.

Step 7   Click **common** and enter the following information:

- **split-as-flows-on-ingress**—Select the traffic aggregation strategy for external ASNs.
- **asn**—Enter the ASN of the internal AS in the network.
- **address-family**—Select the protocol version to include in IAS flows and demands computation.
- **number-of-threads**—Enter the maximum number of threads to be used in parallel computation.
- **ext-node-tags**—Enter a list of one or more node tags separated by a comma.
- **extra-aggregation**—Enter a list of aggregation keys separated by a comma.
- **log-level**—Select the log level of the tool.

Step 8   Click **ias-flows** and enter the following information:

- **ias-computation-timeout-in-minutes**—Enter the timeout for IAS flows computation, in minutes.
- **trim-inter-as-flows**—Enter the value in MBits/sec below which the inter-as-flows for traffic is strictly discarded.
- **match-on-bgp-external-info**—Select whether to match egress IP addresses in the BGP peer relation.
- **flows-dir**—Enter the directory containing flow matrix files to import. The file will be removed immediately after imported.
- **flows-file**—Enter the file path containing flow matrix files to import. The file will be removed immediately after imported.
- **ingress-interface-flow-filter**—Enter a filter of node and interface in the form Node:InterfaceName that will be applied while reading the flow matrix to filter in only those ingress interfaces.
- **egress-interface-flow-filter**—Enter a filter of node and interface in the form Node:InterfaceName that will be applied while reading the flow matrix to filter in only those egress interfaces.
- **backtrack-micro-flows**—Select whether to generate files showing a relationship between micro flows from the input file and those demands or inter-as-flows that aggregate them.

• **flow-import-flow-ids**—Enter comma separated flow IDs to import data from. Use " to import from all flows.

Step 9 Click **demands** and enter the following information:

   • **demand-name**—Enter a namee for any new demands.
   • **demand-tag**—Enter a tag for any new demands, or to be appended to existing tag demands.
   • **trim-demands**—Specify the value in MBits/sec below which the demands are strictly discarded.
   • **service-class**—Specify the demand service class.
   • **traffic-level**—Specify the demand traffic level.
   • **missing-flows**—Enter the path where file with interfaces that are missing flows is generated.

Step 10 Click **run-netflow-collection** > **Invoke run-netflow-collection**

DNF NetFlow Configuration Workflow

To configure DNF and start collection:

Procedure

Step 1 Confirm that the Distributed NetFlow Requirements, on page 117 are met.

Step 2 Configure DNF Cluster, on page 118

   a) Deploy the DNF Cluster using Ansible, on page 118

Step 3 Configure DNF Collection, on page 120

   a) Configure flow_collector_ias and flow_collector_dmd, on page 120
   b) Configure the external-executable-nimo for DNF, on page 120

Distributed NetFlow Requirements

For system requirements, see the *Cisco WAE System Requirements* document.

In addition, the following are required for all cluster elements (controller node, agents, JMS Broker):

   • Agent system requirements meet the same requirements needed for WAE installation.

   • WAE Planning software must be installed on a server (installation server) with the appropriate license file.

   • Routers must be configured to export flows and establish BGP peering with the flow collection server. The flow collection process supports IPv4 and IPv6 flows captured and exported by routers in the ingress direction.

   • Ansible 2.9.18 or later based on python3.

   • Java virtual machine (JVM) has the same installation path for all elements (controller node, agents, JMS Broker). The java executable should be in the path readable for all users.

- A sudo capable, SSH capable user with the same name in each server dedicated for the cluster (broker, controller node, and all the agents) must exist. Make a note of this user name because it is used in the group_vars/all Ansible file (discussed later in this section).

Licensing

Confirm with your Cisco WAE representative that you have the correct licenses for getting flow and flow demands when using the `flow_cluster_controller`, `flow_collector_ias`, and `flow_collector_dmd` tools.

Configure DNF Cluster

Deploy the DNF Cluster using Ansible

Note
- Execute the following steps only from installation server (instance where wae is installed). Ansible takes care of installation, setting up of agents, starting, etc.

- The WAE executable binary file must be present in the installation server.

Procedure

Step 1　Install Ansible version 2.9.18 or higher based on python3. Use the following command:

```
sudo yum install ansible
```

Step 2　Inside WAE directory, modify the `etc/netflow/ansible/hosts` file to include agent, broker, controller node IP addresses. To do this, replace <element-x> with IP addresses/server name as needed.

For example,

For 3 agents and 1 controller node (broker usually resides in the controller node), you must add two more lines to the default DNF agent-1.

Note　Cisco does not recommend using `ansible_ssh_pass`. Instead `use --ask-pass` while running ansible-playbook commands.

Sample Config:

```
[dnf-broker]
10.10.10.1 ansible_ssh_user={{TARGET_SSH_USER}} ansible_ssh_pass={{SSH_USER_PASS}}

[dnf-controller]
10.10.10.1 ansible_ssh_user={{TARGET_SSH_USER}} ansible_ssh_pass={{SSH_USER_PASS}}

[dnf-agent-1]
10.10.10.2 ansible_ssh_user={{TARGET_SSH_USER}} ansible_ssh_pass={{SSH_USER_PASS}}

[dnf-agent-2]
10.10.10.3 ansible_ssh_user={{TARGET_SSH_USER}} ansible_ssh_pass={{SSH_USER_PASS}}

[dnf-agent-3]
10.10.10.4 ansible_ssh_user={{TARGET_SSH_USER}} ansible_ssh_pass={{SSH_USER_PASS}}
```

Step 3   Modify the `package/linux/wae/etc/netflow/ansible/startup.yml` file to include/uncomment/add agents as needed to match the number of agents/controller node address(es) in `package/linux/wae/etc/netflow/ansible/hosts`.

Step 4   Modify the `package/linux/wae/etc/netflow/ansible/bash/service.conf` file. Change the `<jms-broker-server-name>` to controller node's IP address (In the above sample config, IP address=10.10.10.1).

Step 5   Modify the `package/linux/wae/etc/netflow/ansible/group_vars/all` file. Change all the relevant variables as referenced/used. See group_vars/all, on page 121

       Note   You can add the following line if needed, but Cisco does not recommend the same:

```
SSH_USER_PASS: "ciscowae"
```

Step 6   Export `ANSIBLE_HOME=${WAE_ROOT}/etc/netflow/ansible`.

Use

```
ansible-playbook -i ${ANSIBLE_HOME}/hosts ${ANSIBLE_HOME}/install.yml
```

or

```
ansible-playbook -i ${ANSIBLE_HOME}/hosts ${ANSIBLE_HOME}/install.yml --ask-pass --ask-become-pass
```

depending on whether SSH_USER_PASS is set or not.

Step 7   Prepare the OS for netflow. Use the following command:

```
ansible-playbook -i ${ANSIBLE_HOME}/hosts ${ANSIBLE_HOME}/prepare-agents.yml --ask-pass
--ask-become-pass
```

Step 8   Startup the cluster. Use the following command:

```
ansible-playbook -i ${ANSIBLE_HOME}/hosts ${ANSIBLE_HOME}/startup.yml --ask-pass
```

Step 9   Use the following command to list the cluster elements (optional).

```
ansible-playbook -i ${ANSIBLE_HOME}/hosts ${ANSIBLE_HOME}/list.yml --ask-pass
```

Step 10   Send cluster-config to the cluster. Use the following command:

```
${WAE_ROOT}/bin/flow_cluster_manage -action send-cluster-configuration \
-options-file ${ANSIBLE_HOME}/bash/service.conf \
-cluster-config-file-path <path-of-config>/flow-config-cluster.json
```

       Note   JAVA_HOME needs to be set.

       Example:

export JAVA_HOME=/usr/java_latest

See Create the DNF Cluster Configuration File, on page 122 to create `<path-of-config>`/flow-config-cluster.json.

Step 11   Use the following command to check cluster status (optional):

```
${WAE_ROOT}/bin/flow_cluster_manage -action request-cluster-status \
-options-file ${ANSIBLE_HOME}/bash/service.conf
```

Shutdown/Uninstall the DNF Cluster

To shutdown the cluster, use the following command:

```
ansible-playbook -i ${ANSIBLE_HOME}/hosts ${ANSIBLE_HOME}/shutdown..yml --ask-pass
```

To uninstall, use the following command:

```
ansible-playbook -i ${ANSIBLE_HOME}/hosts ${ANSIBLE_HOME}/uninstall.yml --ask-pass
```

Configure DNF Collection

Configure flow_collector_ias and flow_collector_dmd

These CLI tools are configured inside the
`<WAE_installation_directory>`/etc/netflow/ansible/bash/nimo_flow_collector_ias_dmd.sh script
and is executed within the external-executable-nimo. The `flow_collector_ias` and `flow_collector_dmd`
tools generate demands and demand traffic with NetFlow data received from the cluster. Edit the as follows:

Before editing, change the permissions on this file:

chmod +x nimo_flow_collector_ias_dmd.sh

- **CUSTOMER_ASN**—Enter ASN.

- **SPLIT_AS_FLOWS_ON_INGRESS**—When multiple external ASNs are connected to an IXP switch,
  it determines whether to aggregate traffic from all ASNs or to distribute it proportionally to MAC
  accounting ingress traffic. The default value is aggregate. The other value is mac-distribute.

- **ADDRESS_FAMILY**—Enter list of protocol versions to include (comma-separated entries). The default
  is ipv4,ipv6.

- **WAIT_ON_CLUSTER_TIMEOUT_SEC**—Enter the number of seconds to wait before for timing out
  when delegating the computation of the IAS flows into the distributed cluster. The default is 60 seconds.

`nimo_flow_collector_ias_dmd.sh` example:

```
#!/bin/bash

# this script should be called from NSO's 'external executable NIMO' configuration window
# in this way:
# /path-to/nimo_flow_collector_ias_and_dmd.sh $$input $$output

# modify as needed - BEGIN
CUSTOMER_ASN=142313
SPLIT_AS_FLOWS_ON_INGRESS=aggregate
ADDRESS_FAMILY=ipv4,ipv6
WAIT_ON_CLUSTER_TIMEOUT_SEC=60
# modify as needed - END
```

For more information on `flow_collector_ias` or `flow_collector_dmd`options, navigate to
wae-installation-directory/bin and enter **flow_collector_ias -help** or **flow_collector_dmd
-help**.

Configure the external-executable-nimo for DNF

The external-executable-nimo runs the `nimo_flow_collector_ias_dmd.sh` script against a selected network
model. In this case, you take an existing model created in WAE and append information from
`nimo_flow_collector_ias_dmd.sh` to create a final network model that contains the flow data you want.

Before you begin

- You must have a source network model. This is the final network model which includes topology collection
  and any other NIMO collections you want to include.

• Confirm that you have already completed the preliminary tasks in DNF NetFlow Configuration Workflow, on page 117.

Procedure

Step 1 From the Expert Mode, navigate to **/wae:networks**.

Step 2 Click the plus (+) sign and enter a network model name. We recommend a unique name that is easily identifiable; for example, networkABC_CNF_flow_get.

Step 3 Click the **nimo** tab.

Step 4 From the **Choice - nimo-type** drop-down list, choose **external-executable-nimo**.

Step 5 Click **external-executable-nimo** and select the source network.

Step 6 Click the **advanced** tab and enter the following:

• **argv**—Enter <directory_path>/nimo_flow_collector_ias_dmd.sh $$input $$output.

Step 7 To verify configuration, click **run** from the external-executable-nimo tab.

Note For aggregation, under aggregator configuration, add the `external-executable-nimo` nimo network as a dependency of type netflow-nimo.

Create the Ansible Configuration Files

If you use default WAE installation options, there are only a few mandatory parameters that must be changed. These will be noted in the applicable configuration topics. The topics described in this section assume the following:

• The controller node server (installation server) is where the WAE planning software has been installed and default directories are used. In particular, the configuration files used for DNF on the installation server are located in `<wae_installation_directory>/etc/netflow/ansible`.

• A dedicated JMS broker will be used in DNF configuration.

• In configuration examples, the following values are used:

• Controller node and JMS broker IP address—198.51.100.10

• Agent 1 IP address—198.51.100.1

• Agent 2 IP address—198.51.100.2

• Agent 3 IP address—198.51.100.3

group_vars/all

The file is located in `<WAE_installation_directory>/etc/netflow/ansible/group_vars/all`. This file is the Ansible file that contains the variable definitions that are used in the playbook files.

Edit the following options:

Option	Description
LOCAL_WAE_INSTALLATION_DIR_NAME	The local path that contains the WAE installation file.
WAE_INSTALLATION_FILE_NAME	The filename of the WAE installation file.
TARGET_JDK_OR_JRE_HOME	The full path and filename of the Oracle JRE file. All machines in the cluster (broker, primary node, and all the agents) should have the JRE previously installed under this variable.
LOCAL_LICENSE_FILE_PATH	The full path to the license file.
SSH_USER_NAME	The SSH user name created or used when SSH was enabled on each machine. This sudo user is used by Ansible to deploy the cluster over SSH.

For example (comments removed):

```
LOCAL_WAE_INSTALLATION_DIR_NAME: "/wae/wae-installation"
WAE_INSTALLATION_FILE_NAME: "wae-linux-v7.4.0.bin"
TARGET_JDK_OR_JRE_HOME: "/usr/lib/jvm/jre-11-openjdk-11.0.7"
LOCAL_LICENSE_FILE_PATH: "/home/user1/.cariden/etc/MATE_Floating.lic"
TARGET_SSH_USER: ssh_user
```

Create the DNF Cluster Configuration File

To create the cluster configuration file for CNF/DNF, you can use `flow_manage` with `-action produce-cluster-config-file` option, and edit the json file as required.

For example:

Procedure

Step 1 Use the following sample file to create the .json file.

Source the `waerc` file.

```
${CARIDEN_HOME}/flow_manage \
-action produce-cluster-config-file \
-node-flow-configs-table <input-path> \
-cluster-config-file <output-path> \
-interval 120 \
-bgp true \
-bgp-port 10179 \
-port 12100 \
-flow-size lab \
-server-ip ::
```

where `<input-path>` is the path of the `node-flow-configs-table` created under Create the node-flow-configs-table File, on page 112 and `<output-path>` is the path where you want the resulting seed cluster configuration file to reside.

A sample `<input-path>` file would look like this:

```
<NodeFlowConfigs>
Name    SamplingRate    FlowSourceIP    BGPSourceIP    BGPPassword
node1   1024    192.168.75.10    69.127.75.10    ag5Xh0tGbd7
node2   1024    192.168.75.11    69.127.75.11    ag5Xh0tGbd7
```

Verify that the output of the seed cluster configuration file is similar to the following:

```
{
  "agentConfigMapInfo": {
    "cluster_1::instance_x": {
      "perAgentDebugMode": null,
      "flowManageConfiguration": {
        "maxBgpdPeers": 150,
        "useBgpPeering": true,
        "outfileProductionIntervalInSecs": 120,
        "networkDeploymentSize": "lab",
        "bgpTcpPort": 10179,
        "netflowUdpPort": 12100,
        "daemonOutputDirPath": "<user.home>/.cariden/etc/net_flow/flow_matrix_interchange",
        "keepDaemonFilesOnStart": false,
        "keepDaemonFilesOnStop": true,
        "purgeOutputFilesToKeep": 3,
        "routerConfigList": [
          {
            "name": "node1",
            "bGPSourceIP": "69.127.75.10",
            "flowSourceIP": "192.168.75.10",
            "bGPPassword": "ag5Xh0tGbd7",
            "samplingRate": "1024"
          },
          {
            "name": "node2",
            "bGPSourceIP": "69.127.75.11",
            "flowSourceIP": "192.168.75.11",
            "bGPPassword": "ag5Xh0tGbd7",
            "samplingRate": "1024"
          }
        ],
        "ipPrefixFilteringList": [],
        "appendedProperties": null,
        "daemonOutputFileMaskPrefix": "out_matrix_",
        "daemonOutputSoftLinkName": "flow_matrix_file-latest",
        "extraAggregation": [],
        "listValidExtraAggregationKeys": false
      }
    },
    "cluster_1::instance_y": {
      "perAgentDebugMode": null,
      "flowManageConfiguration": {
        "maxBgpdPeers": 150,
        "useBgpPeering": true,
        "outfileProductionIntervalInSecs": 120,
        "networkDeploymentSize": "lab",
        "bgpTcpPort": 10179,
        "netflowUdpPort": 12100,
        "daemonOutputDirPath": "<user.home>/.cariden/etc/net_flow/flow_matrix_interchange",
        "keepDaemonFilesOnStart": false,
        "keepDaemonFilesOnStop": true,
        "purgeOutputFilesToKeep": 3,
        "routerConfigList": [
          {
            "name": "node1",
            "bGPSourceIP": "69.127.75.10",
            "flowSourceIP": "192.168.75.10",
            "bGPPassword": "ag5Xh0tGbd7",
            "samplingRate": "1024"
          },
          {
            "name": "node2",
```

```
              "bGPSourceIP": "69.127.75.11",
              "flowSourceIP": "192.168.75.11",
              "bGPPassword": "ag5Xh0tGbd7",
              "samplingRate": "1024"
          }
        ],
        "ipPrefixFilteringList": [],
        "appendedProperties": null,
        "daemonOutputFileMaskPrefix": "out_matrix_",
        "daemonOutputSoftLinkName": "flow_matrix_file-latest",
        "extraAggregation": [],
        "listValidExtraAggregationKeys": false
      }
    }
  },
  "aggregationMode": "okIfNotAllPortionsArePresent",
  "debugMode": {
    "bypassAnyNfacctdOperation": false
  },
  "logNetflowTraffic": false
}
```

Step 2 Edit the file to include each agent configuration. Copy, paste, and edit each section as it applies to each agent in the cluster. This example shows two agents:

```
{
  "agentConfigMapInfo": {
    "cluster_1::instance_x": {
      "perAgentDebugMode": null,
      "flowManageConfiguration": {
        "maxBgpdPeers": 150,
        "useBgpPeering": true,
        "outfileProductionIntervalInSecs": 120,
        "networkDeploymentSize": "lab",
        "bgpTcpPort": 10179,
        "netflowUdpPort": 12100,
        "daemonOutputDirPath": "<user.home>/.cariden/etc/net_flow/flow_matrix_interchange",
        "keepDaemonFilesOnStart": false,
        "keepDaemonFilesOnStop": true,
        "purgeOutputFilesToKeep": 3,
        "routerConfigList": [
          {
            "name": "node1",
            "bGPSourceIP": "69.127.75.10",
            "flowSourceIP": "192.168.75.10",
            "bGPPassword": "ag5Xh0tGbd7",
            "samplingRate": "1024"
          },
          {
            "name": "node2",
            "bGPSourceIP": "69.127.75.11",
            "flowSourceIP": "192.168.75.11",
            "bGPPassword": "ag5Xh0tGbd7",
            "samplingRate": "1024"
          }
        ],
        "ipPrefixFilteringList": [],
        "appendedProperties": null,
        "daemonOutputFileMaskPrefix": "out_matrix_",
        "daemonOutputSoftLinkName": "flow_matrix_file-latest",
        "extraAggregation": [],
        "listValidExtraAggregationKeys": false
      }
    },
```

```
    "cluster_1::instance_y": {
      "perAgentDebugMode": null,
      "flowManageConfiguration": {
        "maxBgpdPeers": 150,
        "useBgpPeering": true,
        "outfileProductionIntervalInSecs": 120,
        "networkDeploymentSize": "lab",
        "bgpTcpPort": 10179,
        "netflowUdpPort": 12100,
        "daemonOutputDirPath": "<user.home>/.cariden/etc/net_flow/flow_matrix_interchange",
        "keepDaemonFilesOnStart": false,
        "keepDaemonFilesOnStop": true,
        "purgeOutputFilesToKeep": 3,
        "routerConfigList": [
          {
            "name": "node1",
            "bGPSourceIP": "69.127.75.10",
            "flowSourceIP": "192.168.75.10",
            "bGPPassword": "ag5Xh0tGbd7",
            "samplingRate": "1024"
          },
          {
            "name": "node2",
            "bGPSourceIP": "69.127.75.11",
            "flowSourceIP": "192.168.75.11",
            "bGPPassword": "ag5Xh0tGbd7",
            "samplingRate": "1024"
          }
        ],
        "ipPrefixFilteringList": [],
        "appendedProperties": null,
        "daemonOutputFileMaskPrefix": "out_matrix_",
        "daemonOutputSoftLinkName": "flow_matrix_file-latest",
        "extraAggregation": [],
        "listValidExtraAggregationKeys": false
      }
    }
  },
  "aggregationMode": "okIfNotAllPortionsArePresent",
  "debugMode": {
    "bypassAnyNfacctdOperation": false
  },
  "logNetflowTraffic": false
}
```

Note   The .json file configuration is only needed for Controller node and is not required for Processor node.

Multi WAE Collection

Table 4: Feature History

Feature	Release Information	Description
Multi WAE Collection Support	Cisco WAE Release 7.5.0	WAE is now updated to support Multi WAE Collection. This is very usefule for large networks. Multi WAE allows you to split the WAE topology into smaller WAE instances. Collection is done in parallel for each WAE instance and plan file generated from each instance is then merged into a single plan file for the entire network.

This section contains the following topics:

Multi WAE Collection Overview

Multi WAE collection allows you to split the WAE topology into smaller WAE instances. Collection is done in parallel for each WAE instance. The plan file generated after collection from each WAE instance is then merged to a single plan file for the entire network.

Multi WAE is particularly useful for large networks. It improves performance and reduces the collection time by performing collection and model building of few NIMOs in parallel using different WAE instances. Any change in the network is propagated to that particular WAE instance final network using notification. The merge NIMO, when run will propagate to the final merged network.

Each WAE instance has its own WAE process running with access to WAE CLI and WAE GUI. WAE Design can also be installed in each of the WAE instances and can be used to view the plan files from each instance.

The SR-PCE (XTC) agent is configured in active state only on one WAE instance and this is called the WAE Scale Primary instance. SR-PCE (XTC) agent is disabled for other WAE instances and these are called WAE Scale Secondary instances. Each WAE instance is given a unique scale ID which is shared between the HA primary and HA secondary instances. This ID is called the Scale ID.

Example:

If a network has around 3k devices within the single SR-PCE (XTC) server, then Multiple WAE instances can be configured to process a subnetwork of the topology. Here the split of the network is done by SR-PCE (XTC) agent.

The BGPLS XTC NIMO which is the first NIMO to run after SR-PCE (XTC) agent is configured to read the sub network database file instead of full SR-PCE (XTC) agent output database. BGPLS XTC NIMO processes the sub network and provides input to other NIMOs. The remaining NIMOs work on the sub network. Output from SAGE contains topology and traffic information pertaining to the sub network. Merge NIMO merges the output from each WAE instance SAGE network.

NIMOs like Netflow NIMO, Multicast NIMO, and Demands NIMO require full network as source network and hence must be run after the merged NIMO. It is recommended to have a separate DARE and SAGE workflow starting with merged NIMO as the only source and the remaining required dependency NIMO like Demands, etc as dependency network.

For multiple AS typologies, recommended way is to configure separate WAE instances for each AS and then merge their SAGE plan files to get full network topology. A non multi WAE approach is recommended when multiple AS topology is being discovered. It is simpler and has no dependency on multi WAE.

Splitting WAE into Multiple Instances

You can split WAE topology into multiple instances based on area, level, AS, node count, or other user defined configurations. The maximum number of supported WAE instances is 5.

Split Based on Area/Level/AS

The split is based on the area/level/AS and the number of nodes coming under the area/level/AS.

- If number of areas/levels/AS is equal to the number of WAE instances participating in the split, then one WAE instance processes nodes belonging to one area/level/AS.

- If number of areas/levels/AS is more than the number of WAE instances then one or more areas/levels/AS are grouped and associated to each WAE instance depending on the node count belonging to the area/level/AS.

- If the number of areas/levels/AS is less than the number of WAE instances then the area/level/AS having more than the desired node count [= total node count/ number of WAE instances] is split into one or more WAE instances.

Split Based on Node Count

The split is purely based on the node count only. The total node count divided by the number of WAE instances is put into each of the WAE instances. The reminder nodes are added to the WAE instance belonging to the lowest Scale ID.

Split Based on User Defined Configurations

You can specify the WAE topology to split based on some user defined configurations. Comma separated IP Addresses, Host names, Areas, Levels, or AS for each Scale ID can be specified and the same will be used for the split.

Note IP addresses provided by SR PCE agent are used. IP Manage is not used for the split.

Merging Topologies

The plan files from each of the WAE instances are merged by the **inter-as-nimo**. The following information is merged.

- All the topology related tables like Nodes, Circuits, Interfaces, BGP, SRLG, etc

- All the LSP related tables like LSPs, SegmentLists, NamedPaths, etc

- Traffic information

Note
- It is recommended to schedule the merge NIMO to enable propagation of the topology updates through notification to the final merged network.

- The L2VPN information is not merged since the same VPN is named in each of the split files.

Licensing

The license for Multi WAE works similar to the HA setup. In case of smart licensing, secondary users must be added for additional WAE instances taking part in Multi WAE. In case of traditional licensing, the license file must have list of MAC address associated with additional WAE instances taking part in Multi WAE.

Multi WAE Collection Workflow

The following workflow describes the high-level steps to configure multi WAE collection when using the Cisco WAE CLI.

> **Note** There can be a maximum 5 WAE instances participating in Multi WAE. There can be a maximum of 1 scale primary instance.

Before you begin

Install Multi WAE using the ansible playbook. See *Cisco WAE Installtion Guide*.

Procedure

Step 1 Configure XTC agent on all servers. See Configure Agents Using the Expert Mode, on page 28.

Step 2 Configure multi WAE on all servers.

Step 3 Restart XTC (on multi WAE scale primary server).

> **Note** On all multi WAE secondary servers XTC agent will be disabled.

Step 4 Configure other NIMOs on all severs – **topo-bgpls-xtc-nimo**, **lsp-pcep-xtc-nimo**, **lsp-snmp-nimo**, **topo-vpn-nimo**, **topo-bgp-nimo**, **traffic-poll-nimo**, **inventory-nimo**. See Network Interface Modules (NIMOs), on page 55.

> **Note** Multicast nimo and **layout-nimo** are not supported on individual servers due to partial topology.

Step 5 Configure **inter-as-nimo** on multi WAE scale primary which merges SAGE plan of all servers to get final topology.

Step 6 Configure final network on on multi wae scale primary instance. Configure **inter-as-nimo** as the source for final network. Add **layout-nimo**, **multicast nimo**, and **traffic-demands-nimo** to the final network.

Step 7 Run collection of all NIMO on all servers. Run **inter-as-nimo** on scale primary server.

Step 8 Verify and merge the final plan file.

Cisco WAE UI—Multi WAE

You can use Cisco WAE UI to access the details related to different WAE instances and to see the split details.

> **Note** The Multi WAE Dashboard option is available only if Multi WAE is configured in WAE.

Navigate to **Cisco WAE UI** > **Multi WAE Dashboard**.

This is a view-only screen providing details about the number of splits, split type, username, install, run path, etc. You can use this screen to view the IP addresses of remote WAE and remote HA WAE as well.

You can check the status of the WAE instances participating in Multi WAE using **Cisco WAE UI** > **Status Dashboard**. See Status Dashboard, on page 176.

Cisco WAE CLI—Multi WAE

The different configuration options available for multi WAE are:

ha-enabled	Indicates if HA is configured for the WAE instances taking part in the split.
igp-protocol	IGP protocol running in the network.
install-path	Install path for the WAE instance (mandatory parameter).
num-of-splits	Number of WAE instances configured to handle network split (default value = 2).
remote-wae-details	WAE details of all the WAE instances **Note** The number of entries in **remote-wae-details** must be equal to **num-of-splits**, else commit does not succeed.
run-path	Run directory for the WAE instance (mandatory parameter).
split-enabled	If set, the WAE instance is part of Multi-WAE instance.
split-type	This field indicates the split algorithm to be used to split the topology.
user-name	User name associated with the WAE instance (mandatory parameter).
remote-wae-ha-details	IP address of the WAE HA instance associated with the scale ID.

Note   Once the Multi WAE configuration is done in all the WAE instances, the changes done to basic multi WAE configuration (other than the advanced configuration) in scale primary instance is propagated to all the scale secondary instances using Kafka.

To set advanced options, use:

```
wae components multi-wae advanced
```

Options:

asn	Autonomous system number for the network.
debug-health-check	Action added for debug purpose only.

health-check-enabled	Specify if you want to enable the health check. Default is false.
health-check-interval	Specify the interval between the health check runs in minutes. Default is 5 min and minimum is 1 min.
kafkaPort	Kafka port exposed by scale primary.
load-split-config	Loads the split config from yang configuration.
node-filter	Use a node filter.
node-split-record-file-path	The record file path to store the mapping between the node and split number when record is enabled.
record-node-split-mapping	Enable recording of the node to split mapping.
rsync-pool-size	The number of threads processing rsync tasks, default is 5.
rsync-timeout	Rsync Timeout for fetching xtc topology output from primary wae instance in minutes, default is 20 minutes.
single-ended-ebgp-discovery	Discover eBGP links that only have a single link end (not common).
split-action	Can be used for debug purpose. Split the agent topology db file into sub files.

Use the following WAE CLI commands to retrieve information related to Multi WAE.

- To retrieve details of every WAE instance taking part in the split, use:

  ```
  wae components multi-wae remote-wae-details <scale-id>
  ```

- To see the Multi WAE config used by WAE, use:

  ```
  wae components multi-wae load-split-config
  ```

  Use the command to see the different multi WAE configuration values used internally in WAE instance in **wae-java-vm.log** file. This command can be used for debugging purposes.

- To verify how the split output looks for a given Multi WAE configuration, use:

  ```
  wae components multi-wae split-action
  ```

- To clear the Multi WAE config, use:

  ```
  wae components multi-wae clear-multi-wae-config
  ```

- To copy the configuration from scale primary to scale secondary given the xpath, use:

  ```
  wae components multi-wae copy-config
  ```

Note
- The multi WAE configuration must be first configured on all the instances for this command to work.

- **copy-config** works on scale primary instance only.

Health Check in Multi WAE

You can check the status of the WAE instances participating in Multi WAE using Health Check. An Ansible playbook is used to determine the status of the WAE instances. The playbook is used to get the status which is displayed on Status Dashboard.

Configure Multi WAE so that all the prerequisites required for running of the playbook are met:

```
wae components multi-wae advanced health-check
```

Options

health-check-enabled	Specify if you want to enable the health check. Default is false.
health-check-interval	Specify the interval between the health check runs in minutes. Default is 5 min and minimum is 1 min.

If health check enabled, the **topo-bgpls-xtc-nimo** or **lsp-pcep-xtc-nimo** cannot be run until health check service completes. Wait for the scheduled interval to run or run the below command to enable running of the **topo-bgpls-xtc-nimo**.

```
wae components multi-wae get-remote-wae-health run-xtc-status-check true
```

It is not mandatory to run the health check on the scale secondary instances unless HA is configured for scale primary. The health check status collected by scale primary is propagated to all the scale secondary instances via kafka.

High Availability in Multi WAE

Every WAE instance participating in Multi WAE can have the HA standby instance. Configure HA using the following command:

```
wae components multi-wae ha-enabled true remote-ha-wae-details
```

When HA is enabled on multi WAE primary instance, Health Check must be enabled on scale primary. When HA is enabled on multi WAE secondary instance, Health Check must be enabled on all servers. For more information, see .

Multi WAE Configuration Examples

- Sample Multi WAE Configuration

```
show running-config wae components multi-wae
 wae components multi-wae split-enabled true
wae components multi-wae ha-enabled false
wae components multi-wae num-of-splits 2
wae components multi-wae user-name user1
wae components multi-wae run-path /home/user1/750/mw/wae-run/
wae components multi-wae install-path /home/user1/750/mw/wae-install/
wae components multi-wae igp-protocol ospf
wae components multi-wae remote-wae-details 11
 ip-address 10.0.0.1
 role       scale-primary
!
wae components multi-wae remote-wae-details 14
 ip-address 10.0.0.8
 role       scale-secondary
!
wae components multi-wae split-type area
wae components multi-wae advanced health-check-enabled true
```

- Sample Merge Nimo Configuration

```
networks network merge_nimo
 nimo status active  false
 nimo status last-run 2021-08-23T07:06:14.933+00:00
 nimo status last-successful-run 2021-08-23T07:06:14.933+00:00
 nimo inter-as-nimo single-as-merge true
 nimo inter-as-nimo sources wae-redhat-1
  network        sage
 wae-scale-id   11
 !
 nimo inter-as-nimo sources wae-redhat-2
  network        sage
 wae-scale-id   14
 !
!
```

- Sample Final network with demands configuration

```
wae components aggregators aggregator final_dare
 sources source merge_nimo
  nimo inter-as-nimo
 !
 dependencies dependency traffic_demand
  nimo traffic-demands-nimo
 !
 final-network final_sage
!
```

- Sample show command output

```
show wae components multi-wae split-details
       SPLIT
SCALE  TYPE
ID     VALUES
---------------
11     0,40,30
14     20,10
```

- Sample show Health Status output

```
show wae components multi-wae health-status
wae components multi-wae health-status last-run "23-Aug-21 12:30:40 IST"
wae components multi-wae health-status active-topo-xtc-agents XTC-Standby,XTC-Active
wae components multi-wae health-status active-lsp-xtc-agents XTC-Standby,XTC-Active
wae components multi-wae health-status wae-details 10.0.0.1
 scale-id                 11
```

```
role                     scale-primary
wae-status               true
kafka-status             true
ha-status                Disabled
topo-xtc-agents-configured  XTC-Standby,XTC-Active
lsp-xtc-agents-configured   XTC-Standby,XTC-Active
primary-scale-id-configured 11
primary-scale-ip-configured 10.0.0.1
wae-version              "WAE v7.5.0-822-g95e355c for linux on x86_64."
wae components multi-wae health-status wae-details 10.0.0.1
scale-id                 14
role                     scale-secondary
wae-status               true
kafka-status             true
ha-status                Disabled
topo-xtc-agents-configured  XTC-Standby,XTC-Active
lsp-xtc-agents-configured   XTC-Standby,XTC-Active
primary-scale-id-configured 11
primary-scale-ip-configured 10.0.0.1
wae-version              "WAE v7.5.0-822-g95e355c for linux on x86_64."
```

Multi WAE Collection Limitations

- XTC agent configuration is required on all servers and is controlled from only the primary server only.

- WAE can be split into a maximum of 5 servers.

- After any change in Multi WAE configuration –

  - Restart the XTC agent

  - Resync Aggregator

- After changing the number of splits, WAE services must be stopped and started again.

- Apply node filter configuration under multi WAE configuration. After applying node filter configuration under multi WAE configuration knob, restart XTC agent.

- L2VPN is not supported in Multi WAE environment.

- Recommended way to remove Multi WAE configuration –

```
#wae components multi-wae clear-multi-wae-config
```

Telemetry Configuration

This section contains the following topics:

Telemetry Overview

Streaming model-driven telemetry (MDT) provides a mechanism to select data of interest from IOS XR routers and to transmit it in a structured format to a collector such as WAE that can use the data for near real-time monitoring and optimization of the network. For more information on MDT, see the "Configure Model-driven Telemetry" chapter in the Cisco IOS XR Telemetry Configuration Guide.

WAE understands IOS XR operational YANG models and can receive streamed telemetry from IOS XR routers, then parse and store the data. After the data is in WAE, the sr-traffic-matrix NIMO can read the data and use it to create a network model with demands.

Configure Telemetry in WAE

Before you begin

Enable segment routing for routers and configure traffic collector on devices to make sure there is traffic in the system. Check for prefix and tunnel traffic on the routers using the following commands:

Prefix Traffic:

```
sh traffic-collector ipv4 counters prefix <prefix-name>
```

Tunnel Traffic:

```
show traffic-collector ipv4 counters tunnels <tunnel-name>
```

Procedure

Step 1   Configure the WAE Telemetry Agent.

```
admin@wae(config)# wae agents telemetry-agent ports <port-number1> <port-number2> <port-numberxx>
admin@wae(config)# commit
```

Note   The ports that the agent will use to receive telemetry information must be available on the WAE machine.

Step 2   Configure telemetry on devices to send Key-Value Google Protocol Buffers (KV-GPB) encoded telemetry on ports using TCP that will be used for WAE. Three attributes must be defined on the router: sensor-group, destination-group, and the subscription. For information on how to do this, see the "Configure Model-driven Telemetry" chapter in the Cisco IOS XR Telemetry Configuration Guide. Examples are provided at the end of this procedure.

Step 3   Configure the sr-traffic-matrix-nimo. For more information, see Segment Routing Traffic Matrix Collection, on page 65.

```
admin@wae(config)# networks network <network-model-name> nimo sr-traffic-matrix-nimo source-network
 <source-network> collection-period <collection-period-in-seconds>
```

`collection-period` configuration is enabled by default and is set to 60 sec.

The options available are:

```
admin@wae(config)# networks network srt nimo sr-traffic-matrix-nimo ?
Possible completions:
  advanced
  collection-period   Frequency in seconds for automatic periodic generation of demands ('0' value
disables periodic demand generation).
  source-network      Source network for this network to use.
  <cr>

admin@wae(config)# networks network srt nimo sr-traffic-matrix-nimo advanced ?
Possible completions:
  action-timeout            Specifies the timeout value (in minutes) for running actions - default
 of '0' specifies the system default.
  copy-network              When set to 'true', copies the source network into this NIMO network
and create demands in the new model.
  telemetry-agent-callback  Callback for telemetry-agent to inform sr-traffic-matrix-nimo about new
 telemetry data.
```

`telemetry-agent-callback` is not an configuration option and it is an action used by Telemetry Agent internally.

Note   For SR policy (XTC-nimo), `use-signaled-name` property in the source network has to be set to true (by default it is true) while performing collection.

For RSVP LSP tunnels (lsp-snmp-nimo), `use-signaled-name` property in the source network has to be set to false (by default it is false) while performing LSP collection.

Step 4   Run the sr-traffic-matrix-nimo collection to generate demands.

```
admin@wae# networks network <network-model-name> nimo sr-traffic-matrix-nimo run-collection
```

By default, demands are generated by using locally cached information. However, if you want demands to be generated using the raw telemetry data from the WAE Telemetry agent, you must set the **use-cache** option to false. For example:

```
admin@wae# networks network <network-model-name> nimo sr-traffic-matrix-nimo run-collection use-cache
 false
```

Example

1. Configure the WAE Telemetry Agent

```
admin@wae# config terminal
Entering configuration mode terminal
admin@wae(config)# wae agents telemetry-agent ports 1624
admin@wae(config)# commit
```

2. Configure the router to send telemetry data to WAE:

a. Define sensor-group

```
telemetry model-driven
 sensor-group SRTM
  sensor-path Cisco-IOS-XR-infra-tc-oper:traffic-collector/afs/af/counters/tunnels
  sensor-path
Cisco-IOS-XR-infra-tc-oper:traffic-collector/vrf-table/default-vrf/afs/af/counters
 !
!
```

b. Define destination-group

```
telemetry model-driven
 destination-group my_workstation
  address-family ipv4 10.152.130.41 port 1624
   encoding self-describing-gpb
   protocol tcp
 !
```

Note The ip-address and port from the above example must be the same as the one configured previously in WAE telementry agent.

c. Define subscription

```
telemetry model-driven
 subscription ABC
  sensor-group-id SRTM sample-interval 5000
  destination-id my_workstation
 !
!
```

3. Configure the SR LSP Traffic Matrix NIMO (sr-traffic-matrix-nimo)

```
admin@wae# config terminal
Entering configuration mode terminal
admin@wae(config)# networks network srtm nimo sr-traffic-matrix-nimo source-network igp
collection-period 50s
admin@wae(config)# commit
```

To view connections between WAE and the router, use the shell CLI netstat command. For example:

```
# netstat -an | grep :1624 | grep ESTABLISHED
tcp        0     28 10.10.10.10:1624          10.152.130.41:61092      ESTABLISHED
```

where 10.10.10.10 is the address of the WAE machine and 10.152.130.41 is the address of the connected router.

4. Run the sr-traffic-matrix-nimo collection to generate demands.

```
admin@wae# networks network srtm nimo sr-traffic-matrix-nimo run-collection
status true
```

```
message Succeeded: Retrieved 12 SR demands from network srtm
admin@wae# show running-config networks network srtm model demands | nomore
networks network igp
model demands demand "PE1|PE2|default"
source node node-name PE1
destination node node-name PE2
service-class-name default
traffic 22.203833
!
.......
!
model demands demand "PE4|PE3|default"
source node node-name PE4
destination node node-name PE3
service-class-name default
traffic 22.202989
!
!
admin@wae#
```

CHAPTER 11

Automation Applications

This section contains the following topics:

Automation Applications

Automation applications rely on having real-time network models to work with. Applications get a copy of the latest network model (primary model) to either run optimization against or manipulate based on what the application's purpose or function is.

You can then view the network model using WAE Design (**File** > **Open From** > **WAE Modeling Daemon**).

Bandwidth on Demand Configuration Workflow

The Bandwidth on Demand application models and predicts the impact of a new service. This application is used when provisioning new services that requires persistent bandwidth and specific IGP or TE metric demands. The application finds a path for the SR policies being delegated in the network. For more information on the Bandwidth on Demand application, see Bandwidth on Demand Application, on page 5.

This workflow describes the high-level configuration steps to configure Bandwidth on Demand and other components.

Note

- Before enabling Bandwidth on Demand, confirm that the Bandwidth Optimization application is not running. You cannot run both applications at the same time.

- Make sure the sr-traffic-matrix-nimo as part of the aggregator (step 4) is enabled. See Segment Routing Traffic Matrix Collection, on page 65

Steps	For more information, see ...
1. Configure device authgroups and SNMP groups	Configure Device Access Using the Expert Mode, on page 27.
2. Configure a network access profile	Configure Network Access, on page 28
3. Configure the XTC agent	Configuring XTC Agents Using the Expert Mode, on page 28
4. Configure the aggregator	NIMO Collection Consolidation, on page 62 **Note** Make sure sr-traffic-matrix-nimo is part of the configuration.
5. Configure the WAE Modeling Daemon (WMD)	Configure the WAE Modeling Daemon (WMD), on page 93
6. Run topology and additional NIMOs.	Network Interface Modules (NIMOs), on page 55
7. Configure the Bandwidth on Demand application and SR policies.	Configure Bandwidth on Demand, on page 142
8. Open the network model.	You can get a visual network model layout using WAE Design. From WAE Design, navigate to **File** > **Open From** > **WAE Modeling Daemon** and select the final network model. **Note** After initial configuration, you may run NIMOs anytime and the Bandwidth on Demand application will update the network model.

Configure Bandwidth on Demand

Before you begin

This procedure describes the configuration options for the Bandwidth on Demand application. For the complete configuration workflow, see Bandwidth on Demand Configuration Workflow, on page 141. For an example of the entire workflow, see Initial Bandwidth on Demand CLI Configuration Example, on page 143

Procedure

Step 1   From the Expert Mode, in **Configuration editor**, navigate to **/wae:wae/components/bw-on-demand:bw-on-demand** and click the **config** tab.

Step 2   Enter the following values:

- **xtc-agents**—Select the XTC agents.
- **enable**—Select true to enable the Bandwidth on Demand application.
- **keepalive**—Enter the keepalive interval. The Bandwidth on Demand application and XTC exchange keepalive messages to maintain a persistent connection. If the connection fails, the Bandwidth on Demand application shuts down, clears current state, tries to reconnect, and re-delegate the SR policies.

- **priority**—In case there are multiple Bandwidth on Demand application instances, enter the priority level for this instance. XTC will then delegate instances depending on the priority level.
- **util-threshold**—Enter the congestion constraint (in percentage).When the Bandwidth on Demand application searches a path for the policies being delegated, it will avoid any paths that may exceed the congestion utilization threshold.
- **reopt-interval**—Enter the duration after which the LSPs must be re-optimized.
- **metric-reopt-interval**—Enter the duration after which the LSPs must be re-optimized for metric.
- **priority-mode**—Select true to enable priority mode.

Note For **advanced** options, contact your Cisco WAE representative.

Step 3 Click **Commit** to save the configuration.

Step 4 Configure a new SR policy with bandwidth and IGP or TE metric type on a device. See the applicable Cisco IOS XR documentation for your specific device configuration (for example, "Configure SR-TE Policies"). Device configuration example:

```
segment-routing
   traffic-eng
       policy BWOD_2TO3_IGP
           bandwidth 10000
           color 100 end-point ipv4 198.51.100.3
           candidate-paths
               preference 10
                   dynamic mpls
                       pce
                           address ipv4 198.51.100.1
                           exit
                       metric
                           type igp
```

Step 5 Open the resulting network model using WAE Design (**WAE Design** > **File** > **Open From** > **WAE Modeling Daemon**).

Initial Bandwidth on Demand CLI Configuration Example

The following is an example of an initial Bandwidth on Demand CLI configuration within a Cisco Virtual Internet Routing Lab (VIRL) test environment. After initial configuration, you may run NIMOs anytime and the Bandwidth on Demand application will update the network model.

Configure device and network discovery.

```
# config
# devices authgroups group virl_test default-map
# devices authgroups group virl_test default-map remote-name cisco
# devices authgroups group virl_test default-map remote-password cisco
# devices authgroups group virl_test default-map remote-secondary-password cisco
# devices authgroups snmp-group virl_test default-map
# devices authgroups snmp-group virl_test default-map community-name cisco
# wae nimos network-access network-access virl_test default-auth-group virl_test
# wae nimos network-access network-access virl_test default-snmp-group virl_test
# wae nimos network-access network-access virl_test node-access 198.51.100.1 auth-group
virl_test
# wae nimos network-access network-access virl_test node-access 198.51.100.1 snmp-group
virl_test
# wae nimos network-access network-access virl_test node-access 198.51.100.1 ip-manage
192.0.2.131
# wae nimos network-access network-access virl_test node-access 198.51.100.2 auth-group
virl_test
```

```
# wae nimos network-access network-access virl_test node-access 198.51.100.2 snmp-group
virl_test
# wae nimos network-access network-access virl_test node-access 198.51.100.2 ip-manage
192.0.2.132
# wae nimos network-access network-access virl_test node-access 198.51.100.3 auth-group
virl_test
# wae nimos network-access network-access virl_test node-access 198.51.100.3 snmp-group
virl_test
# wae nimos network-access network-access virl_test node-access 198.51.100.3 ip-manage
192.0.2.133
# wae nimos network-access network-access virl_test node-access 198.51.100.4 auth-group
virl_test
# wae nimos network-access network-access virl_test node-access 198.51.100.4 snmp-group
virl_test
# wae nimos network-access network-access virl_test node-access 198.51.100.4 ip-manage
192.0.2.134
# wae nimos network-access network-access virl_test node-access 198.51.100.5 auth-group
virl_test
# wae nimos network-access network-access virl_test node-access 198.51.100.5 snmp-group
virl_test
# wae nimos network-access network-access virl_test node-access 198.51.100.5 ip-manage
192.0.2.135
# wae nimos network-access network-access virl_test node-access 198.51.100.6 auth-group
virl_test
# wae nimos network-access network-access virl_test node-access 198.51.100.6 snmp-group
virl_test
# wae nimos network-access network-access virl_test node-access 198.51.100.6 ip-manage
192.0.2.136
# wae nimos network-access network-access virl_test node-access 198.51.100.7 auth-group
virl_test
# wae nimos network-access network-access virl_test node-access 198.51.100.7 snmp-group
virl_test
# wae nimos network-access network-access virl_test node-access 198.51.100.7 ip-manage
192.0.2.137
```

Configure XTC agents.

```
# wae agents xtc xtc virl enabled xtc-host-ip 192.0.2.131
```

Configure the BGP network (topo-bgpls-xtc-nimo).

```
# networks network virl_bgpls nimo topo-bgpls-xtc-nimo xtc-host virl igp-protocol isis
extended-topology-discovery true backup-xtc-host virl network-access virl_test advanced
nodes remove-node-suffix virl.info
```

Configure the LSP PCEP network (lsp-pcep-xtc-nimo).

```
# networks network virl_pcep_lsp nimo lsp-pcep-xtc-nimo xtc-hosts virl xtc-host virl
# networks network virl_pcep_lsp nimo lsp-pcep-xtc-nimo source-network virl_bgpls advanced
 sr-use-signaled-name true
```

Set the network that the aggregator will write to.

```
# networks network virl_final_model
```

Configure continuous polling (traffic-poll-nimo).

```
# networks network virl_cp nimo traffic-poll-nimo network-access virl_max source-network
virl_dare iface-traffic-poller enabled
# networks network virl_cp nimo traffic-poll-nimo lsp-traffic-poller enabled
# networks network virl_cp nimo traffic-poll-nimo advanced snmp-traffic-population
```

```
scheduler-interval 0
```

Configure the aggregator to subscribe to source networks.

```
# wae components aggregators aggregator virl_final_model sources source virl_bgpls
# wae components aggregators aggregator virl_final_model sources source virl_pcep_lsp
```

Configure WMD. In this example, WMD is set up to run demand mesh and demand deduction for all applications using WMD. So, when the continuous poller updates WMD, WMD triggers demand deduction.

```
# wae components wmd config network-name virl_final_model dare dare-destination
virl_final_model
# wae components wmd config network-name virl_final_model demands add-demands true
demand-mesh-config dest-equals-source true
```

Run NIMOs (network collection).

```
networks network virl_bgpls nimo topo-bgpls-xtc-nimo run-xtc-collection
networks network virl_pcep_lsp nimo lsp-pcep-xtc-nimo run-collection
```

Configure Bandwidth on Demand.

```
# configure
# wae components bw-on-demand config xtc-host 192.0.2.131 xtc-port 8080 util-threshold 90.0
# wae components bw-on-demand config advanced lsp-traffic max-simulated-requested
primary-objective min-metric private-new-lsps true
# commit
# exit
```

Open base network model using WAE Design (**WAE Design** > **File** > **Open From** > **WAE Modeling Daemon**) to compare the resulting network model (after SR policies have been configured and the Bandwidth on Demand application has been executed).

Configure SR policies on a device.

```
# configure
# segment-routing
# traffic-eng
# policy BWOD_2TO3_IGP
# bandwidth 1000
# color 100 end-point ipv4 192.0.2.132
# candidate-paths
# preference 10
# dynamic mpls
# pce
# address ipv4 192.0.2.130
# exit
# metric
# type igp
# commit
# end
```

After the SR policy configuration is committed, WMD is updated and the Bandwidth on Demand application calculates the best path given the congestion restraint and IGP metric. Open the resulting network model using WAE Design (**WAE Design** > **File** > **Open From** > **WAE Modeling Daemon**) to compare the baseline network model to the new network model.

Shut Down Bandwidth on Demand

To properly shut down the Bandwidth on Demand application, the following steps must be done in order:

Procedure

Step 1   Stop Bandwidth on Demand.

```
# wae components bw-on-demand config enable false
# commit
```

Step 2   Stop the WAE Modeling Daemon.

```
# wae components wmd config enable false
# commit
```

Step 3   Stop XTC agents.

```
# wae agents xtc xtc <network_name> disable xtc-host-ip <xtc_ip_address>
# commit
```

Bandwidth Optimization Application Workflow

Table 5: Feature History

Feature	Release Information	Description
Collection of Multiple SR Candidate Paths using SR-PCE	Cisco WAE Release 7.5.0	WAE now processes all candidate paths from an XTC agent. For an lsp-pcep-xtc-nimo, WAE processes all candidate paths during data collection and path addition/deletion during notification.

The Bandwidth Optimization application is designed to react and manage traffic as the state of the network changes. It determines if any changes in the network state will cause congestion. If so, the Bandwidth Optimization application computes the LSPs and sends them to XTC for deployment.

WAE processes all candidate paths from an XTC agent. For an **lsp-pcep-xtc-nimo**, WAE processes all candidate paths during data collection and path addition/deletion during notification.

This workflow describes the high-level configuration steps needed to configure the Bandwidth Optimization application and other components.

Note

- Make sure the sr-traffic-matrix-nimo as part of the aggregator (step 4) is enabled. See Segment Routing Traffic Matrix Collection, on page 65

Steps	For more information, see ...
1. Configure device authgroups and SNMP groups	Configure Device Access Using the Expert Mode, on page 27.
2. Configure a network access profile	Configure Network Access, on page 28
3. Configure the XTC agent	Configuring XTC Agents Using the Expert Mode, on page 28
4. Configure the aggregator	NIMO Collection Consolidation, on page 62 **Note** Make sure sr-traffic-matrix-nimo is part of the configuration.
5. Configure the WAE Modeling Daemon (WMD)	Configure the WAE Modeling Daemon (WMD), on page 93
6. Run topology and additional NIMOs.	Network Interface Modules (NIMOs), on page 55
7. Configure the Bandwidth Optimization application	Configure Bandwidth Optimization, on page 147
8. Open the network model	You can get a visual network model layout using WAE Design. From WAE Design, navigate to **File** > **Open From** > **WAE Modeling Daemon** and select the final network model. **Note** After initial configuration, you may run NIMOs anytime and the Bandwidth on Demand application will update the network model.

Configure Bandwidth Optimization

This procedure describes the configuration options for the Bandwidth Optimization application. For the complete configuration workflow, see Bandwidth Optimization Application Workflow, on page 146.

Procedure

Step 1   From the Expert Mode, in **Configuration editor**, navigate to **/wae:wae/components/bw-opt** and click the **config** tab.

Step 2   Enter the following values:

- **xtc-agents**—Select the XTC agents.
- **enable**—Select true to enable the Bandwidth Optimization application.
- **util-threshold**—Enter a percentage that must be exceeded if optimization is to occur. The default is 100%.

- **util-hold-margin**—Enter the margin below the util-threshold interface utilization must be to remove an existing tactical SR policy.
- **color**—Color representing the SR policy for XTC. For more information, contact your Cisco WAE representative.
- **del-lsps**—Select true to delete all app created tactical SR policies when the app is disabled.
- **max-global-reopt-interval**—Enter the time interval to re-optimize the existing tactical SR policies globally.

Step 3   Click **Commit** to save the configuration.

Step 4   After the tool runs, you can click **created-lsps** to view the SR LSPs that wer created for optimized routing..

Step 5   Open the resulting network model using WAE Design (**WAE Design** > **File** > **Open From** > **WAE Modeling Daemon**).

Example

CLI (in config mode) example:

```
# wae components bw-opt config color 2000 enable false threshold 90 xtc-host 192.0.2.131
xtc-port 8080
```

WAE SR Policy Limitations

When using the latest IOS-XR SR features associated with the SR Policy, the following WAE limitations exist:

- If an SR LSP is created through WAE, a default color will be set to an SR LSP.

- Multiple LSPs cannot have the same color, source, and destination.

Shut Down Bandwidth Optimization

To properly shut down the Bandwidth Optimization application, the following steps must be done in order:

Procedure

Step 1   Stop Bandwidth Optimization.

```
# wae components bw-opt config enable false
# commit
```

Step 2   Stop the WAE Modeling Daemon.

```
# wae components wmd config enable false
# commit
```

Step 3   Stop XTC agents.

```
# wae agents xtc xtc <network_name> disable xtc-host-ip <xtc_ip_address>
# commit
```

CHAPTER 12

Scheduler Configuration

This section provides examples and instructions on how to schedule cron and subscription jobs.

Scheduler Overview

The Scheduler performs two types of scheduling jobs:

- Cron job—A time-based job scheduler that allows certain operations to run at a specific date and time. For example, you can schedule collections to run periodically.

- Subscription job—An event-based job scheduler that listens to event notifications defined by triggers from specified sources; for example, network model changes that are pushed to the Scheduler.

Configure the Scheduler

This procedure describes how to schedule cron and subscription-based jobs using the Expert Mode.

Note Scheduler configurations using the Expert Mode or Cisco WAE CLI will not appear in the Cisco WAE UI. To schedule network collections and agents to run using the Cisco WAE UI, see Schedule Jobs Using the Network Model Composer, on page 23.

Before you begin

Configuration of any dependent actions or events must be completed before adding them to the Scheduler.

Procedure

Step 1 From the Expert Mode, in **Configuration editor**, navigate to **wae:wae** > **components** tab > **scheduler** > **task**.

Step 2   Click the plus (+) icon and enter a Scheduler job name.

Step 3   Click **Add**.

Step 4   Define what action to take when the Scheduler job is enabled:

     a) From the **action** tab, click the plus (+) icon and enter the action name.

     b) Click **Add**.

     c) From the **Choice-action** drop-down list, choose **rpc**.

     d) Click **rpc** and enter the path name. The path name designates the operation to invoke. For example, to invoke a network collection, enter the following path:

        /wae:networks/network{<*network_model_name*>}/nimo/<*nimo_name*>/run-collection

     e) (Optional) If certain parameters must be met to invoke the action, click the **params** tab and add the parameters in order of requirement.

        Value for RPC params of type **ENUMERATION** must be specified with a fully qualified path.

        Example:

```
admin@wae# show running-config wae components scheduler tasks task filter
wae components scheduler tasks task filter
action filter
  order 1
  rpc path /wae:networks/network{node-filter}/opm/node-filtering:node-filtering/run
  rpc params 1
   key   delimiter
  value /wae:networks/network{node-filter}/opm/node-filtering:node-filtering/run/delimiter/COMMA

   type  ENUMERATION
  !
```

Step 5   Identify what type of event(s) trigger the action. (If there are multiple triggers, the action is executed if any trigger is invoked.)

     a) From the **trigger** tab, click the plus (+) icon and enter the trigger name.

     b) Click **Add**.

     c) From the **Choice-trigger-spec** drop-down list, choose the trigger type: **cron** or **subscription**.

Step 6   If you are configuring a subscription-based job, click the **subscription** link and do the following:

     a) Enter the source path of the trigger. For example, if the source of the event is a circuit change, enter

        /wae:networks/network{<*network_model_name*>}/model/circuits/circuit

     b) From the **subscription-type** drop-down list, choose one of the following:

        • **operational**—Applies to any operational (read-only) changes, such as traffic polling. These changes are not user initiated.

        • **configuration**—Applies to any configuration changes (addition, deletion, or modification in the network), such as user-initiated LSP configuration changes.

Step 7   If you are configuring a cron-based job, click the **cron** link and enter the applicable parameters that define when to run the action.

Step 8   To add more triggers, repeat the preceding steps. (If there are multiple triggers, the action is executed if any trigger is invoked.)

Step 9   Click **Commit**.

Configure a Trigger for Topology Collection to Run Example

This example configures a subscription-based job that triggers the topology collection to run. The following procedure configures the Scheduler to run BGP-LS collection when a change occurs in the network model. For more information, see Topology Collection Using XTC, on page 59.

Procedure

Step 1 From the Expert Mode, in **Configuration editor**, navigate to **wae:wae** > **components** tab > **scheduler** > **task**.

Step 2 Click the plus (+) icon and enter **run-topo-bgpls** as the Scheduler job name.

Step 3 Click **Add**.

Step 4 Define what action to take when the Scheduler job is enabled:

a) From the **action** tab, click the plus (+) icon and enter **run-xtc-topo** as the action name.

b) Click **Add**.

c) From the **Choice-action** drop-down list, choose **rpc**.

d) Click **rpc** and enter the path name. The path name designates the operation to invoke. For example, to invoke a network collection, enter the following path:

`/wae:networks/network{NetworkABC_topo-bgpls-xtc-nimo}/nimo/topo-bgpls-xtc-nimo/run-collection.`

Step 5 Identify what type of event triggers this action:

a) From the **trigger** tab, click the plus (+) icon and enter **xtc-objects** as the trigger name.

b) Click **Add**.

c) From the **Choice-trigger-spec** drop-down list, choose **subscription**.

Step 6 Click the **subscription** link and do the following:

a) Enter the source path (in this example, it is where the XTC link status changes):

`/wae/agents/xtc/xtc{TTE-xtc11}/pce/xtc-topology-objects/xtc-links`

b) From the **subscription-type** drop-down list, choose **operational**.

Step 7 Click **Commit**.

Example

If using the WAE CLI (in config mode), enter:

```
# wae components scheduler tasks task run-topo-bgpls action run-xtc-topo rpc path
"/wae:networks/network{NetworkABC_topo-bgpls-xtc-nimo}/nimo/topo-bgpls-xtc-nimo/run-xtc-collection"
# wae components scheduler tasks task run-topo-bgpls triggers trigger xtc-objects subscription
 node "/wae/agents/xtc/xtc{TTE-xtc11}/pce/xtc-topology-objects/xtc-links"
# wae components scheduler tasks task run-topo-bgpls triggers trigger xtc-objects subscription
 subscription-type operational
# commit
```

Cisco Smart Licensing

Cisco WAE supports both Cisco Smart Licensing and traditional licensing. If you would like to convert from a traditional license to Smart Licensing, see your Cisco WAE account representative. For information on the differences between the two types of licensing, refer to the Cisco Smart Licensing Overview on Cisco.com.

Note License for different functional packs are provided separately, and is not integrated with Cisco WAE Smart License.

A license is required to use all the features in Cisco WAE. If you have questions about obtaining a license, contact your Cisco support representative or system administrator.

Note If you decide to move back from Smart License to traditional license, disable Smart License from Cisco WAE UI and manually delete the file `MATE_Smart.lic` file present in `~/.cariden/etc` folder.

The tasks mentioned in this chapter can be performed by a user with administrative capabilities.

Before configuring High Availability (HA), configure Smart License on both the setups. Follow the below mentioned steps to configure Smart license on both the systems.

This section contains the following topics:

Cisco Smart Licensing Overview

Cisco offers Smart Licensing, which enables you to monitor Cisco WAE software licenses and endpoint license consumption easily and efficiently through a simple registration and license consumption reporting

process instead of having to install node-locked license files. Details of all Cisco products and licenses that you have purchased are maintained in a centralized database called the Cisco Smart Software Manager (CSSM).

Smart Licensing Configuration Workflow

Procedure

Step 1 Create a Smart Account with Cisco Systems. To do this, go to Smart Account Request and follow the instructions on the web site.

Step 2 Do one of the following

- If using WAE Design, enable Smart Licensing using the WAE Design GUI or CLI in your local WAE Design setup (Windows, Mac, or Linux). For more information, see *WAE Design GUI Installation Guide*.

- If using WAE Collector Server, after sourcing waerc, run

```
license_install -smart-lic-host <hostname or IP> -smart-lic-port 2022 -smart-lic-username admin
 -smart-lic-password Admin@123
```

where

`-smart-lic-host` is the hostname or IP of machine on which WAE server is running (the place where WAE is installed and all nimos are running).

`-smart-lic-port` is the port on which the WAE server listens for netconf messages. Unless you have modified, this should be 2022

`-smart-lic-username` is the user name to login to WAE server. Unless you have modified, this should be admin

`-smart-lic-password` is the password to login to WAE server. Unless you have modified, this should be Admin@123

After the command is run, `MATE_Smart.lic` file will be created under `~/.cariden/etc` folder.

Step 3 Register Cisco WAE with the Cisco Smart Software Manager, on page 156

Step 4 (Optional) By default, the Smart License information is sent directly to the cloud. To change this, follow the steps outlined in Configure the Transport Mode Between Cisco WAE and the CSSM, on page 155

Enable Smart Licensing in Cisco WAE

Before you begin

Confirm that you have a Smart Account. If not, go to Smart Account Request and follow the instructions on the web site.

Note You can disable Smart Licensing at any time by clicking **Disable Smart Software Licensing** from **WAE UI > Smart Licensing**.

Procedure

Step 1   From WAE UI, click Smart Licensing.

Step 2   Click **Enable Smart Software Licensing**.

Step 3   Click **OK** to confirm that you are enabling Smart Licensing. The Smart Licensing page appears.

Step 4   Do one of the following:

a) If you have not registered Cisco WAE with the CSSM on Cisco.com, Cisco WAE will be in evaluation mode (which expires in 90 days). Register Cisco WAE as described in Register Cisco WAE with the Cisco Smart Software Manager, on page 156.

b) If you have registered Cisco WAE with the CSSM, select the licenses you want to use.

Note   To enable Smart Licensing for WAE Design, see the *Cisco WAE Design GUI Installation Guide*.

Configure the Transport Mode Between Cisco WAE and the CSSM

By default, the Smart License information directly to the cloud. To change this, perform the following steps:

Procedure

Step 1   From the Smart Software Licensing page, click **View/Edit** from the Transport Settings field. The Transport Settings window appears.

Step 2   Select a communication mode:

- Direct mode—Send license information directly to the cloud. This is the default. You cannot edit this URL.
- Transport Gateway—Use a Cisco Call Home transport gateway or a Cisco Smart Licensing Software satellite. (A satellite is installed on customer premises and provides a subset of CCSM functionality. See Smart Licensing for more information.

  If you are using satellite for registration, make sure to install 7-202001 version of the satellite.

- HTTP Proxy—Use an HTTP/HTTPS proxy for communication between and the cloud. Enter the proxy IP Address, Port, Username and Password.

Step 3   Click **Save**.

Register Cisco WAE with the Cisco Smart Software Manager

To register Cisco WAE with the Cisco Smart Software Manager (CSSM), you must obtain a token from the CSSM and enter it into the WAE UI. This is a one-time requirement. If Cisco WAE is already registered, you must deregister the product, then register it again. For information on how to use the CSSM, including renewing license registration and authorization, refer to the Cisco Smart Software Manager User Guide. You can also access these operations within the WAE UI using the Smart Licensing page.

Before you begin

- Confirm that you have a Smart Account. If not, go to Smart Account Request and follow the instructions on the web site.

- Confirm that Smart Licensing is enabled in Cisco WAE. See Enable Smart Licensing in Cisco WAE, on page 154.

Procedure

Step 1   Go to the Cisco Software Central web site.

Step 2   Obtain your tokens. If you already have tokens, proceed to the next step.

Step 3   If your token is no longer valid, you can obtain a new token using this procedure.

a) On Cisco Software Central, choose License > Smart Software Licensing.

b) Select the appropriate virtual account.

c) Click the General tab, then click **New Token**.

d) Follow the instructions to provide a name and duration. If you need to create a restricted token to enable export control functions, you can click the **Allow Export-Controlled Functionality** button.

e) Click **Create Token**.

f) Copy the Token ID to your clipboard and proceed to the next step.

Step 4   From the WAE UI, click **Register**. The Smart Software Licensing Product Registration window appears.

Step 5   Enter the Token ID you just copied in Step 3 to register the product instance.

Note   Select the option **Reregister this product instance if it is already registered** to reregister the license.

Step 6   The Smart Software Licensing Status page appears.

Step 7   Click **Choose Licenses...**.

Step 8   Click all applicable licenses from the license table and enter the corresponding number of license instances.

Step 9   Click **OK** to save the changes. The Smart License Usage table is updated with the changes. Click the Refresh icon on the top of the page to manually refresh the data.

Step 10   All details related to the license are available on this screen. The **Actions** drop-down provides further options:

- Renew Authorization Now
- Renew Registration Now
- Reregister
- Deregister

• Disable Smart Software Licensing

Register Cisco WAE in Offline Mode with the Cisco Smart Software Manager

Before you begin

- Confirm that you have a Smart Account. If not, go to Smart Account Request and follow the instructions on the web site.

- Confirm that Smart Licensing is enabled in Cisco WAE. See Enable Smart Licensing in Cisco WAE, on page 154.

- Smart Account License Reservation to be used in offline mode needs specific permission from Cisco. See Smart Licensing for more information.

Note   In Offline mode, only Specific License Reservation (SLR) is supported.

Procedure

Step 1   From WAE UI click Smart Licensing.

Step 2   Click **Register**. The Smart Software Licensing Product Registration window appears.

Step 3   For usage in offline mode, click **Start Here**.

Step 4   Click **Yes, My Smart Account License Reservation Enabled**.

Step 5   On the Smart License Reservation screen, click **Generate Reservation Request Code**. Your License Reservation Request Code is displayed in the following screen. Copy this code using **Copy to Clipboard** button.

Step 6   Go to the Cisco Software Central web site and select the appropriate virtual account.

Step 7   Click Licenses tab, then click **License Reservation**. Paste the License Reservation Request Code here and click **Next**.

Step 8   In the Select Licenses screen, select **Reserve a Specific License** radio button, reserve the necessary licenses from the list and click **Next**.

Step 9   In the Review and Confirm screen, click **Generate Authorization Code**. Copy the code using **Copy to Clipboard** button.

Step 10   Navigate back to WAE UI → Smart Licensing. Click **Enter Reservation Authorization Code**. Paste the Reservation Authorization Code and click **Install Authorization Code/File**.

Step 11   In the WAE UI → Smart Software Licensing screen, use **Choose Licenses...** option to check out necessary number of licenses for usage.

Update Reservation

Use Update Reservation option to reserve more licenses.

Procedure

Step 1   On WAE UI -> Smart Software Licensing screen, make a note of Product Instance Name.

Step 2   Go to the Cisco Software Central web site and select the appropriate virtual account.

Step 3   Click Product Instances tab, search for the name of the product instance that matches the Product Instance Name from WAE UI screen.

Step 4   For this product instance, select **Actions** drop-down and choose **Update Reserved Licenses**.

Step 5   In the Select Licenses screen, select **Reserve a Specific License** radio button, reserve the necessary licenses from the list and click **Next**.

Step 6   In the Review and Confirm screen, click **Generate Authorization Code**. Copy the code using **Copy to Clipboard** button.

Step 7   Navigate back to WAE UI → Smart Licensing. Click **Update Reservation...**. Paste the Reservation Authorization Code and click **Install Authorization Code/File**.

Step 8   A License Reservation Confirmation Code is generated. Copy this code.

Step 9   Navigate back to Cisco Software Central web site. The last step in the Update License Reservation screen is to enter the confirmation code. Click **Enter Confirmation Code**.

Step 10   Enter the Reservation Confirmation Code and click **OK**.

Step 11   In the WAE UI → Smart Software Licensing screen, use **Choose Licenses...** option to check out necessary number of licenses for usage.

Return Reserved Licenses

You can return licenses reserved using Return Reserved Licenses option.

Procedure

Step 1   On WAE UI → Smart Licensing. Click **Return Reserved Licenses...**.

Step 2   In the Confirm Return Licenses screen, click **Generate Reservation Return Code**.

Step 3   Copy the License Reservation Return Code using **Copy to Clipboard** button.

Step 4   Navigate to the Cisco Software Central web site and select the appropriate virtual account.

Step 5   Click Product Instances tab, search for the name of the product instance that matches the Product Instance Name from WAE UI screen.

Step 6   For this product instance, select **Actions** drop-down and choose **Remove**.

Step 7   In the Remove Product Instance pop-up, paste the Reservation Return Code and click **Remove Product Instance**.

Step 8   Close the Remove Product Instance pop-up. You can see that the Registration Status is back to Unregistered state.

Once the Registration Status is back to Unregistered state, the **Disable License Reservation** and **Generate Reservation Request Code** options are available.

Step 9   On WAE UI → Smart Licensing, upon refreshing the page, you can see that the Registration Status is back to Unregistered state.

Smart License Registration and Authorization Statuses

Registration Status

The license registration status reflects whether Cisco WAE is properly registered with Cisco Smart Software Licensing on Cisco.com.

License Registration Status	Description
Unregistered	Smart Software Licensing is enabled on Cisco WAE, but Cisco WAE is not registered with the CSSM.
Registered	Cisco WAE is registered with the CSSM. Cisco WAE has received an ID certificate that will be used for future communication with the Cisco licensing authority. By default, registration renewal happens automatically every 30 days. If you want to manually renew it, click **Renew Registration Now** from the drop-down list available on the top right of the Smart Software Licensing page. To unregister, click **Deregister** from the drop-down list available on the top right of the Smart Software Licensing page.

License Authorization Status

The License Authorization status reflects license usage against purchased licenses, and whether you are in compliance with Cisco Smart Licensing. If you exceed the number of purchased license, the product will be Out of Compliance.

License Authorization Status	Description
Evaluation Mode	Cisco WAE is running in evaluation mode (expires in 90 days).
Authorized	Cisco WAE has a valid Smart Account and is registered. All licenses requested by Cisco WAE are authorized for use.
Out of Compliance	Cisco WAE has exceeded the number of licenses that were purchased.

License Authorization Status	Description
	(Specifically, the virtual account for the product instance has a shortage of one or more licenses types.)
Evaluation Expired	The evaluation period has expired and Cisco WAE is in the unlicensed state.
Authorization Expired	Cisco WAE did not successfully renew its license authorization prior to the authorization expiration date. To renew auhorization, click **Renew Authorization Now** from the drop-down list available on the top right of the Smart Software Licensing page.
Not in Use	A license is not in use.

Administration

This section contains the following topics:

Manage Users

All users have the administrator role. The following procedure describes how to create, modify, and delete users.

Procedure

Step 1 From the WAE UI, click the User Manager icon ( ).

Step 2 To add a user, click and fill in all applicable fields.

Step 3 To change a user's password:

    a) Select the user row and click .

    b) Update the password fields.

    c) Click **Save**.

Step 4 To delete a user, click the user row and click .

Configure Aging

By default, when a circuit, port, node, or link disappears from a network, it is permanently removed and must be rediscovered. To configure how long WAE retains these elements that have disappeared before they are permanently removed from the network, complete the following steps.

Note This is a global option that will be configured for all networks.

Procedure

Step 1 From the Expert Mode, navigate to **/wae:wae/components/aggregators** and click the **aging** tab.

- **aging-enabled**— Select true to enable aging.
- **l3-node-aging-duration**—Enter the time duration for which an L3 node must be kept in the network after it becomes inactive.
- **l3-port-aging-duration**—Enter the time duration for which an L3 port must be kept in the network after it becomes inactive.
- **l3-circuit-aging-duration**—Enter the time duration for which an L3 circuit must be kept in the network after it becomes inactive.

  Note The value of l3-node-aging-duration must be greater than l3-port-aging-duration which inturn must be greater than l3-circuit-aging-duration.

- **l1-node-aging-duration**—Enter the time duration for which an L1 node must be kept in the network after it becomes inactive.
- **l1-port-aging-duration**—Enter the time duration for which an L1 port must be kept in the network after it becomes inactive.
- **l1-link-aging-duration**—Enter the time duration for which an L1 link must be kept in the network after it becomes inactive.

  Note The value of l1-node-aging-duration must be greater than l1-port-aging-duration which inturn must be greater than l1-link-aging-duration.

Step 2 Click **Commit**.

wae.conf

`wae.conf` is an XML configuration file that is formally defined by a YANG model, `tailf-ncsconfig.yang`. This YANG file is included in the WAE distribution, as is a commented `wae.conf.example` file.

The `wae.conf` file controls the baseline of the WAE run time. You can change certain configuration parameters in the `wae.conf` file; for example, you can change the default port that WAE runs on (port 8080) to another port.

> ✎ **Note**   Make sure you restart WAE after you make any changes to `wae.conf` file.

Whenever you start or reload the WAE daemon, it reads its configuration from `./wae.conf` or `<waeruntime-directory>/etc/wae.conf`.

The following example shows `<waeruntime-directory>/etc/wae.conf` contents:

```
<!-- -*- nxml -*- -->
<!-- Example configuration file for wae. -->

<ncs-config xmlns="http://tail-f.com/yang/tailf-ncs-config">

  <!-- WAE can be configured to restrict access for incoming connections -->
  <!-- to the IPC listener sockets. The access check requires that -->
  <!-- connecting clients prove possession of a shared secret. -->
  <ncs-ipc-access-check>
    <enabled>false</enabled>
    <filename>${NCS_DIR}/etc/ncs/ipc_access</filename>
  </ncs-ipc-access-check>

  <!-- Where to look for .fxs and snmp .bin files to load -->

  <load-path>
    <dir>./packages</dir>
    <dir>${NCS_DIR}/etc/ncs</dir>

    <!-- To disable northbound snmp altogether -->
    <!-- comment out the path below -->
    <dir>${NCS_DIR}/etc/ncs/snmp</dir>
  </load-path>

  <!-- Plug and play scripting -->
  <scripts>
    <dir>./scripts</dir>
    <dir>${NCS_DIR}/scripts</dir>
  </scripts>

  <state-dir>./state</state-dir>

  <notifications>
    <event-streams>

      <!-- This is the builtin stream used by WAE to generate northbound -->
      <!-- notifications whenever the alarm table is changed. -->
      <!-- See tailf-ncs-alarms.yang -->
      <!-- If you are not interested in WAE northbound netconf notifications -->
      <!-- remove this item since it does consume some CPU -->
```

```
<stream>
  <name>wae-alarms</name>
  <description>WAE alarms according to tailf-ncs-alarms.yang</description>
  <replay-support>false</replay-support>
  <builtin-replay-store>
    <enabled>false</enabled>
    <dir>./state</dir>
    <max-size>S10M</max-size>
    <max-files>50</max-files>
  </builtin-replay-store>
</stream>

<!-- This is the builtin stream used by WAE to generate northbound -->
<!-- notifications for internal events. -->
<!-- See tailf-ncs-devices.yang -->
<!-- Required for cluster mode. -->
<stream>
  <name>wae-events</name>
  <description>WAE event according to tailf-ncs-devices.yang</description>
  <replay-support>true</replay-support>
  <builtin-replay-store>
    <enabled>true</enabled>
    <dir>./state</dir>
    <max-size>S10M</max-size>
    <max-files>50</max-files>
  </builtin-replay-store>
</stream>

<!-- This is the builtin stream used by WAE to generate northbound -->
<!-- notifications for kicker event stream. -->
<!-- See tailf-kicker.yang -->
<!-- Required for cluster mode. -->
<stream>
  <name>kicker-events</name>
  <description>NCS event according to tailf-kicker.yang</description>
  <replay-support>true</replay-support>
  <builtin-replay-store>
    <enabled>true</enabled>
    <dir>./state</dir>
    <max-size>S10M</max-size>
    <max-files>50</max-files>
  </builtin-replay-store>
</stream>

<!-- This is the builtin stream used by WAE to generate northbound -->
<!-- notifications forwarded from devices. -->
<!-- See tailf-event-forwarding.yang -->
<stream>
  <name>device-notifications</name>
  <description>WAE events forwarded from devices</description>
  <replay-support>true</replay-support>
  <builtin-replay-store>
    <enabled>true</enabled>
    <dir>./state</dir>
    <max-size>S10M</max-size>
    <max-files>50</max-files>
  </builtin-replay-store>
</stream>

<!-- This is the builtin stream used by WAE to generate northbound -->
<!-- notifications for plan state transitions. -->
<!-- See tailf-ncs-plan.yang -->
<stream>
  <name>service-state-changes</name>
```

```
            <description>Plan state transitions according to
            tailf-ncs-plan.yang</description>
            <replay-support>false</replay-support>
            <builtin-replay-store>
              <enabled>false</enabled>
              <dir>./state</dir>
              <max-size>S10M</max-size>
              <max-files>50</max-files>
            </builtin-replay-store>
          </stream>
          <stream>
            <name>XtcNotifications</name>
            <description>Xtc object change notifications</description>
            <replay-support>false</replay-support>
          </stream>
        </event-streams>
    </notifications>

    <!-- Where the database (and init XML) files are kept -->
    <cdb>
      <db-dir>./ncs-cdb</db-dir>
      <!-- Always bring in the good system defaults -->
      <init-path>
        <dir>${NCS_DIR}/var/ncs/cdb</dir>
      </init-path>
    </cdb>


    <!--
        These keys are used to encrypt values of the types
        tailf:des3-cbc-encrypted-string, tailf:aes-cfb-128-encrypted-string
        and tailf:aes-256-cfb-128-encrypted-string.
        For a deployment install it is highly recommended to change
        these numbers to something random (done by WAE "system install")
    -->
    <encrypted-strings>
      <DES3CBC>
        <key1>0123456789abcdef</key1>
        <key2>0123456789abcdef</key2>
        <key3>0123456789abcdef</key3>
        <initVector>0123456789abcdef</initVector>
      </DES3CBC>

      <AESCFB128>
        <key>0123456789abcdef0123456789abcdef</key>
        <initVector>0123456789abcdef0123456789abcdef</initVector>
      </AESCFB128>

      <AES256CFB128>
        <key>0123456789abcdef0123456789abcdef0123456789abcdef0123456789abcdef</key>
      </AES256CFB128>
    </encrypted-strings>


    <logs>
      <syslog-config>
        <facility>daemon</facility>
      </syslog-config>

      <ncs-log>
        <enabled>true</enabled>
        <file>
          <name>./logs/wae.log</name>
          <enabled>true</enabled>
```

```
      </file>
      <syslog>
        <enabled>true</enabled>
      </syslog>
    </ncs-log>

    <developer-log>
      <enabled>true</enabled>
      <file>
        <name>./logs/devel.log</name>
        <enabled>true</enabled>
      </file>
    </developer-log>
    <developer-log-level>error</developer-log-level>

    <audit-log>
      <enabled>true</enabled>
      <file>
        <name>./logs/audit.log</name>
        <enabled>true</enabled>
      </file>
    </audit-log>

    <netconf-log>
      <enabled>true</enabled>
      <file>
        <name>./logs/netconf.log</name>
        <enabled>true</enabled>
      </file>
    </netconf-log>

    <snmp-log>
      <enabled>true</enabled>
      <file>
        <name>./logs/snmp.log</name>
        <enabled>true</enabled>
      </file>
    </snmp-log>

    <webui-access-log>
      <enabled>true</enabled>
      <dir>./logs</dir>
    </webui-access-log>


     <!-- This log is disabled by default if wae is installed using -->
    <!-- the 'system-install' flag. It consumes a lot of CPU power -->
    <!-- to have this log turned on, OTOH it is the best tool to -->
    <!-- debug must expressions in YANG models -->

    <xpath-trace-log>
      <enabled>false</enabled>
      <filename>./logs/xpath.trace</filename>
    </xpath-trace-log>

    <error-log>
      <enabled>true</enabled>
      <filename>./logs/wae-err.log</filename>
    </error-log>

    <progress-trace>
      <enabled>true</enabled>
      <dir>./logs</dir>
    </progress-trace>
```

```
        </logs>

        <ssh>
          <algorithms>
            <kex>diffie-hellman-group14-sha1</kex>
            <mac>hmac-sha2-512,hmac-sha2-256,hmac-sha1</mac>
            <encryption>aes128-ctr,aes192-ctr,aes256-ctr</encryption>
          </algorithms>
        </ssh>

        <aaa>
          <ssh-server-key-dir>${NCS_DIR}/etc/ncs/ssh</ssh-server-key-dir>

          <!-- Depending on OS - and also depending on user requirements -->
          <!-- the pam service value value must be tuned. -->

          <pam>
            <enabled>true</enabled>
            <service>common-auth</service>
          </pam>
          <external-authentication>
            <enabled>false</enabled>
            <executable>$WAE_ROOT/lib/exec/wae-ldap-auth</executable>
          </external-authentication>

          <local-authentication>
            <enabled>true</enabled>
          </local-authentication>

        </aaa>

        <!-- Hash algorithm used when setting leafs of type ianach:crypt-hash, -->
        <!-- e.g. /aaa/authentication/users/user/password -->
        <crypt-hash>
          <algorithm>sha-512</algorithm>
        </crypt-hash>

        <!-- Disable this for performance critical applications, enabling -->
        <!-- rollbacks means additional disk IO for each transaction -->
        <rollback>
          <enabled>true</enabled>
          <directory>./logs</directory>
          <history-size>50</history-size>
        </rollback>


        <cli>
          <enabled>true</enabled>

          <!-- Use the builtin SSH server -->
          <ssh>
            <enabled>true</enabled>
            <ip>0.0.0.0</ip>
            <port>2024</port>
          </ssh>

          <prompt1>\u@wae> </prompt1>
          <prompt2>\u@wae% </prompt2>

          <c-prompt1>\u@wae# </c-prompt1>
          <c-prompt2>\u@wae(\m)# </c-prompt2>

          <show-log-directory>./logs</show-log-directory>
          <show-commit-progress>true</show-commit-progress>
```

```
        <suppress-commit-message-context>maapi</suppress-commit-message-context>
        <suppress-commit-message-context>system</suppress-commit-message-context>
      </cli>

    <webui>
      <absolute-timeout>P1Y</absolute-timeout>
      <custom-headers>
        <header>
          <name>X-Content-Type-Options</name>
          <value>nosniff</value>
        </header>
        <header>
          <name>Content-Security-Policy</name>
          <value>default-src 'self'; script-src 'self'; img-src 'self' data:;
block-all-mixed-content; base-uri 'self'; frame-ancestors 'none'; style-src 'self'
'unsafe-inline'</value>
        </header>
      </custom-headers>
      <idle-timeout>PT30M</idle-timeout>
      <allow-symlinks>true</allow-symlinks>
      <enabled>true</enabled>
      <transport>
        <tcp>
          <enabled>true</enabled>
          <ip>0.0.0.0</ip>
          <port>8080</port>
          <redirect>https://@HOST@:8443</redirect>
          <!-- Uncomment this to enable support for IPv6
          <extra-listen>
            <ip>::</ip>
            <port>8080</port>
          </extra-listen>
          -->
        </tcp>
        <ssl>
          <enabled>true</enabled>
          <ip>0.0.0.0</ip>
          <port>8443</port>
          <key-file>${NCS_DIR}/var/ncs/webui/cert/host.key</key-file>
          <cert-file>${NCS_DIR}/var/ncs/webui/cert/host.cert</cert-file>
          <!-- Uncomment this to enable support for IPv6
          <extra-listen>
            <ip>::</ip>
            <port>8443</port>
          </extra-listen>
          -->
        </ssl>
      </transport>

      <cgi>
        <enabled>true</enabled>
        <php>
          <enabled>false</enabled>
        </php>
      </cgi>
    </webui>

    <rest>
      <enabled>true</enabled>
      <enable-legacy>true</enable-legacy>
    </rest>

    <restconf>
      <enabled>true</enabled>
```

```
        </restconf>

        <netconf-north-bound>
          <enabled>true</enabled>

          <transport>
            <ssh>
              <enabled>true</enabled>
              <ip>0.0.0.0</ip>
              <port>2022</port>
              <!-- Uncomment this to enable support for IPv6
              <extra-listen>
                <ip>::</ip>
                <port>2022</port>
              </extra-listen>
              -->
            </ssh>
            <tcp>
              <enabled>false</enabled>
              <ip>127.0.0.1</ip>
              <port>2023</port>
            </tcp>
          </transport>
        </netconf-north-bound>

        <netconf-call-home>
          <enabled>false</enabled>

          <transport>
            <tcp>
              <ip>0.0.0.0</ip>
              <port>4334</port>
            </tcp>
          </transport>
        </netconf-call-home>

        <!-- <ha> -->
        <!--   <enabled>true</enabled> -->
        <!-- </ha> -->

        <large-scale>
          <lsa>
            <!-- Enable Layered Service Architecture, LSA. This requires
                 a separate Cisco Smart License.
            -->
            <enabled>true</enabled>
          </lsa>
        </large-scale>

      </ncs-config>
```

The default values for many configuration parameters are defined in the YANG file. See wae.conf Configuration Parameters, on page 198.

Configure High Availability

Cisco WAE supports High Availability (HA) with automatic failover. Two instances of WAE nodes are configured to run in parallel, where the primary node is configured in primary mode and the secondary node is configured in standby mode. The primary node listens to the connection from secondary nodes on port 4570 (or the port configured in the wae.conf file). Committed CDB data is mirrored to the secondary node at regular

intervals. Note that any write operations to the CDB (NIMO operations, agent processes, or scheduler actions) which are performed on a node that is in standby mode will fail.

If the primary node fails, the secondary node will takeover as primary node. Once the primary mode is enabled on the secondary node, write operations are allowed, the CDB is rebuilt, and any scheduled jobs will run. It resumes operations that the primary node previously performed.

Procedure

Step 1 On both primary and secondary nodes, edit the wae.conf file to enable HA.

```
<ha>
  <enabled>true</enabled>
  <ip>0.0.0.0</ip>
  <!-- The following port configuration is optional.
       Default port is 4570. This option can be used
       to override the default port -->
  <!-- <port>4570</port> -->
</ha>
```

Note Make sure your /etc/hosts file is updated with the hostname to IP address mapping.

Step 2 Restart Cisco WAE on both nodes using Supervisor

```
sudo supervisorctl restart wae:*
```

Step 3 On both nodes, do one of the following:

• From the Cisco WAE CLI:

```
# wae ha-config nodes n1-name <hostname1>
# wae ha-config nodes n1-address <server-ip1>
# wae ha-config nodes n1-wae-uname <user1>
# wae ha-config nodes n2-name <hostname2>
# wae ha-config nodes n2-address <server-ip2>
# wae ha-config nodes n2-wae-uname <user2>
# wae ha-config cluster-id <cluster-id>
# wae ha-config temp-dir-location <temp-location>
```

On the primary, initiate be-primary and verify status:

```
# wae ha-config be-primary
# wae ha-config status
```

Initiate be-secondary on the secondary node and verify status:

```
# wae ha-config be-secondary
# wae ha-config status
```

Note The temp-dir-location is the path to where archive plan files are copied from primary to secondary location. Hence it is recommended to add path to a directory which has enough space to hold all the archive files while replicating.

• From the WAE UI:

a. Click the HA configuration icon.

Note Make sure your /etc/hosts file is updated with the hostname to IP address mapping

.

b. Enter the N1 Node and N2 Node details.

> **Note** Provide Fully Qualified Domain Name for the Node Name of both nodes.

c. Enter the **Cluster-ID**.

d. Select **be-primary**, **be-secondary** or **be-none**.

e. Click **Secondary**.

> **Note** • Once a node is selected as secondary node, only WAE UI → HA Configuration and WAE UI → Status pages are enabled for that node.
>
> • When moving a node from **be-none** to **be-secondary**, make sure that there are no collections running on the node and that there are no agents/scheduler tasks configured.

Step 4 The status of the node is displayed on the HA Configuration page.

> **Note** After HA is configured on both nodes, any further configuration changes is allowed only on the primary node.

Step 5 Set up passwordless ssh between both the nodes for data sync.

Generate authentication key

```
# ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/user/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/user/.ssh/id_rsa.
Your public key has been saved in /home/user/.ssh/id_rsa.pub.
The key fingerprint is:
1x:x2:x4:22:5a:7x:2x:a5:a5:4x:6x:88:2c:33:x8:77 root@remote-host
```

Copy the public key to the remote host

```
# ssh-copy-id -i ~/.ssh/id_rsa.pub user@remote-host
user@remote-hosts's password:
```

Alternatively if the server is not installed with openssh-clients (a package which provides ssh-copy-id command utility) you can copy the authentication key with the command:

```
# cat ~/.ssh/id_rsa.pub | ssh user@remote-host "cat >> ~/.ssh/authorized_keys"
```

> **Note** In case of failover, the data sync happens in opposite direction and hence passwordless ssh has to be setup for both the nodes.

Step 6 Schedule data replication using Global Scheduler - HA data sync on primary node to replicate files under `<wae-run-directory>/networks`, `<wae-run-directory>/agents` and any archives.

`rsync` utility is required for data replication.

Install `rsync` utility if its is not available in your system using the following command:

```
sudo yum install rsync
```

Note	• Once the primary node goes down, the secondary node comes up as primary. The primary node comes up as None once it is restored. You have to manually configure this node as Secondary again.
	• It is necessary to schedule the run collection for XTC based NIMOs to get the complete network model after failover. If the run collection is not scheduled, XTC based reactive changes will not work until collection is run manually.
	• When a Node is moved from None state to Secondary state, make sure that collection or agents are not running on that node.
	• Netflow workflow and layout-nimo are not supported under HA.

Troubleshoot High Availability

There are two logs that should be looked at when troubleshooting HA configuration:

• *<wae_run_directory>*/logs/wae-java-vm.log

• *<wae_run_directory>*/logs/devel.log

The following table lists HA errors and their meanings.

Error Code	ENUM	Description
25	CONFD_ERR_HA_CONNECT	Failed to connect to a remote HA node.
26	CONFD_ERR_HA_CLOSED	A remote HA node closed its connection to WAE or there was a timeout waiting for a sync response from the primary during a confd_ha_beseecondary() call.
27	CONFD_ERR_HA_BADFXS	A remote HA node has either a different set of FXS ports or different versions of FXS ports compared to WAE.
28	CONFD_ERR_HA_BADTOKEN	A remote HA node has a different token than WAE.
29	CONFD_ERR_HA_BADNAME	A remote HA node has a different name than the name captured in WAE.
30	CONFD_ERR_HA_BIND	There was a failure to bind the HA socket for incoming HA connections.
31	CONFD_ERR_HA_NOTICK	A remote HA node failed to produce the interval live ticks.

Configure LDAP

Cisco WAE supports authentication of foreign users using Lightweight Directory Access Protocol (LDAP).

Before you configure LDAP on WAE:

- You should be familiar with the LDAP directory tree and its contents.

- Install and configure LDAP server and collection details.

- To use LDAPS protocol, get the SSL certificate and add it to a keystore.

  Commands to get and import SSL certificate

  Save the self signed certificate to cert.pem file using the following command:

  ```
  # openssl s_client -connect <ldap-host>:<ldap-ssl-port> </dev/null 2>/dev/null | sed
  -n '/^-----BEGIN/,/^-----END/ { p }' > cert.pem
  ```

  Get the default key-store path using the following command. Typically the default key-store path is /etc/pki/java/cacerts for CentOS 7 with open-jdk

  ```
  # $WAE_ROOT/lib/exec/test-java-ssl-conn <ldap-host> <ldap-ssl-port> 2>1 | grep "trustStore
   is:"
  ```

  Import cert into default key-store using following command

  ```
  # sudo keytool -import -keystore <default-key-store-path> -storepass changeit -noprompt
   -file cert.pem
  ```

To troubleshoot LDAP configuration, view the following logs:

- LDAP configuration log—*<wae_run_directory>*/logs/wae-javavm.log

- LDAP authentication runtime log—*<wae_run_directory>*/logs/wae-ldap-auth.log

Configure LDAP Using the CLI

Before you begin

Confirm prerequisites are met as described in Configure LDAP, on page 173.

Procedure

Step 1   Edit the wae.conf file to enable external authentication.

```
<<external-authentication>
  <enabled>true</enabled>
  <executable>$WAE_ROOT/lib/exec/wae-ldap-auth.sh</executable>
</external-authentication>
```

Step 2   Restart WAE.

```
# wae --start
```

Step 3   Configure LDAP server details using the WAE CLI.

Example: LDAP configuration

```
# wae_cli -u admin
# conf

(config)# wae ldap-config enabled
(config)# wae ldap-config protocol ldap
(config)# wae ldap-config server 10.220.121.47
(config)# wae ldap-config port 389
(config)# wae ldap-config search-base ou=people,dc=planetexpress,dc=com
(config)# wae ldap-config principal-expression "(uid={0})"
(config)# commit
```

Example: LDAP configuration with SSL and the admin user

```
# wae_cli -u admin
# conf

(config)# devices authgroups group ldap-search default-map
(config)# devices authgroups group ldap-search default-map remote-name cn=admin,dc=company,dc=com
(config)# devices authgroups group ldap-search default-map remote-password HelloDolly
(config)# commit

(config)# wae ldap-config enabled
(config)# wae ldap-config protocol ldaps
(config)# wae ldap-config server 10.222.121.48
(config)# wae ldap-config port 636
(config)# wae ldap-config search-base ou=people,dc=company,dc=com
(config)# wae ldap-config principal-expression "(uid={0})"
(config)# wae ldap-config ldap-auth-group ldap-search
(config)# commit
(config)# exit
```

Example: LDAP configuration for MS Active Directory Server

```
# wae_cli -u admin
# conf

(config)# devices authgroups group  ad-user ldap-search default-map
(config)# devices authgroups group  ad-user ldap-search default-map remote-name
CN=waeuser1,CN=Users,DC=woadtest,DC=local
(config)# devices authgroups group  ad-user ldap-search default-map remote-password HelloWAE
(config)# commit

(config)# wae ldap-config enabled
(config)# wae ldap-config protocol ldap
(config)# wae ldap-config server waelab.cisco.com
(config)# wae ldap-config port 389
(config)# wae ldap-config search-base cn=users,dc=woadtest,dc=local
(config)# wae ldap-config principal-expression "(sAMAccountName={0})"
(config)# wae ldap-config ldap-auth-group ad-user
(
(config)# commit
(config)# exit
```

Configure LDAP Using the WAE UI

Before you begin

Confirm prerequisites are met as described in .

Procedure

Step 1 Edit the wae.conf file to enable external authentication.

```
<<external-authentication>
  <enabled>true</enabled>
  <executable>$WAE_ROOT/lib/exec/wae-ldap-auth.sh</executable>
</external-authentication>
```

Step 2 Restart WAE.

Step 3 From the WAE UI, click the LDAP configuration icon (   ).

Step 4 By default, the Enabled toggle switch is on. If not, enable use of the LDAP server for user authentication and toggle the switch on.

Step 5 Enter the LDAP options. See for more information.

Step 6 Click **Save**.

LDAP Configuration Options

Table 6: LDAP Field Descriptions

Field	Description
Protocol	Protocol used to reach the LDAP server. • LDAP—Transmits communication in clear text. • LDAPS—Transmits communication that is encrypted and secure. Default value is LDAP.
Server *<ldap-server>*	LDAP server IP address or FQDN, which is the server's hostname with the DNS domain name appended to the end. FQDN format: *<LDAP_hostname>* . *<domain>* .com
Port	Port used to reach the LDAP server. For unencrypted authentication the default is TCP 389. For encrypted authentication the default is TCP 636. Default value is 389.
Search Base *<ldap-base-ou>*	This is the Distinguished Name of the base search OU for all user accounts that should have permission to login to the WAE Server.

Field	Description
Principal Expression	Default : `LDAP.Principal.Expr:` `(userPrincipalName={0})` ,
	The `{0}` token will be replaced by the user's input for username at the login page.
	The `userPrincipalName=` must match a User Objects' LDAP attribute that identifies the user under the LDAP search base.
	From the LDAP schema, use the User Unique Attribute `uid`.
	The WAE server will search all objects under the LDAP search base tree for:
	`uid=cisco-mate-user1`
	Common alternatives include `userPrincipalName` or `userName` etc.
LDAP Auth Group	Username/password used to perform LDAP search.
	Enable-password field is not used for LDAP. Enter any dummy value while configuring from UI.
Keystore Path	Keystore path when SSL is enabled.
Keystore Pass	Keystore password.

Status Dashboard

There are situations when Cisco WAE suddenly stops working or it crashes. There are times when traffic poller stops working, or there are problems that occur during archive. These issues sometimes are due to WAE system resources being used up. The status dashboard in WAE helps to address such situations by identifying the processes that cause system leaks or processes that completely use the resources.

To access the status dashboard, navigate to Cisco WAE UI, and click **Status Dashboard**.
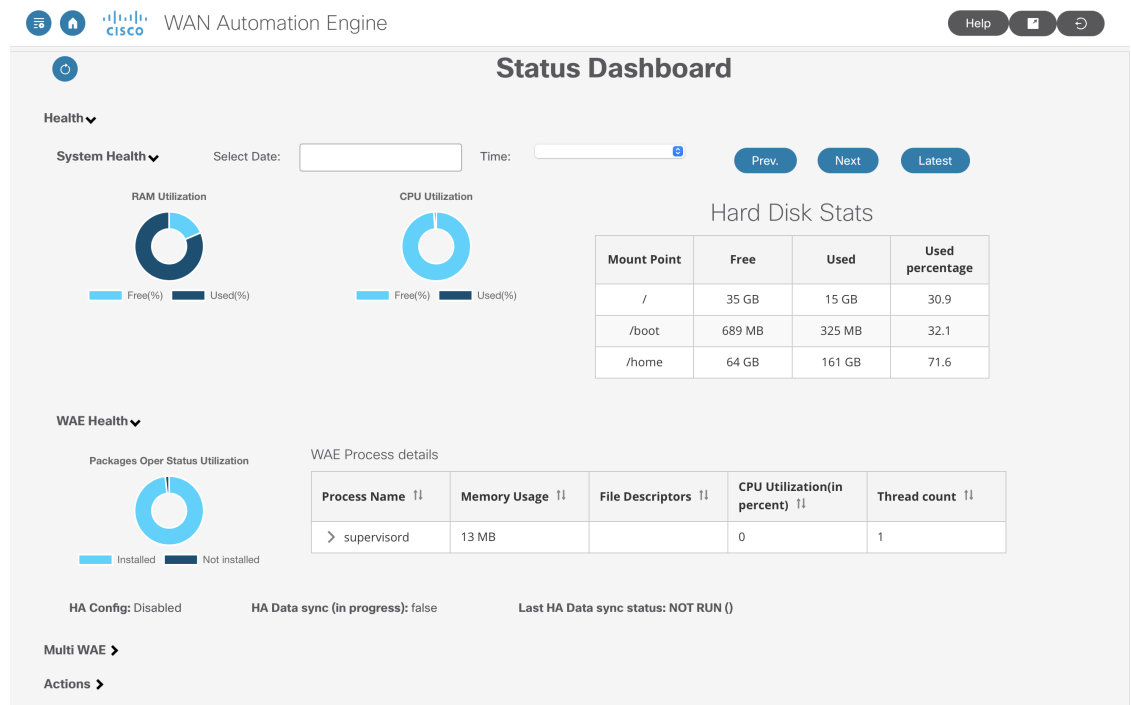
The Status Dashboard is mainly divided into two sections:

- Health

- Actions

Health is further divided into **System Health** and **WAE Health**

System Health captures RAM utilization, CPU utilization and system level hard disk statistics. The tools runs once every 10 minutes and generates the statistics. Use the **Select Date** and **Time** fields to access the required report.

By default, the latest reports are displayed. Use **prev**, **next** buttons to navigate between reports.

The RAM utilization and CPU utilization charts display the used and free spaces. Hover over the used or free areas to read the actual % utilization.

WAE Health captures process level usage details like memory usage, file descriptors and CPU utilization. In case of any issue with a process, access all the relevant process level details using the date and time fields.

There are circumstances wherein after Cisco WAE is deployed, some packages remain uninstalled. **Packages Oper Status Utilization** chart gives the % of installed and uninstalled packages. Click the installed/not installed area to see the list of packages installed/not installed.

HA configuration, if enabled, shows if system is a primary or a secondary.

If Multi WAE is enabled, the health of the WAE instances is also displayed in the Status Dashboard.

Actions section displays the status of NIMO actions and agent actions.

In **NIMO Actions** section, click the NIMO card to get further details of the status of the NIMOs.

| Note | NIMO action displays only current action, not historical action data. |

The **Agent Actions** section displays the action status of a particular agent.

Understand WAE CLI Logging

WAE has extensive logging functionality. WAE logs to the directory specified in the `wae.conf` file. The following are the most useful log files:

- `wae.log`—WAE daemon log; can be configured to syslog.

- `wae_err.log.1,wae_err.log.idx,wae_err.log.siz`—If the WAE daemon has a problem, this log contains debug information for support. Display the content with the command **wae --printlog wae_err.log**.

- `audit.log`—Central audit log that covers all northbound interfaces; can be configured to syslog.

- `localhost:8080.access`—All http requests to the daemon. This is an access log for the embedded web server. This file adheres to the Common Log Format, as defined by Apache and others. This log is disabled by default and is not rotated; that is, use **logrotate(8)**.

- `devel.log`—Debug log for troubleshooting user-written code. This log is enabled by default and is not rotated; that is, use **logrotate(8)**. Use this log with the `java-vm` or `python-vm` logs. The user code logs in the `vm` logs and the corresponding library logs in `devel.log`. Disable this log in production systems. Can be configured to syslog.

- `wae-java-vm.log, wae-python-vm.log`—Log for code running in Java or Python VMs, such as service applications. Developers writing Java and Python code use this log (in combination with `devel.log`) for debugging.

- `netconf.log, snmp.log`—Log for northbound agents; can be configured to syslog.

- `rollbackNNNNN`—All WAE commits generate a corresponding rollback file. You can configure the maximum number of rollback files and file numbering in `wae.conf`.

- `xpath.trace`—XPATH is used in many places, such as XML templates. This log file shows the evaluation of all XPATH expressions. To debug XPATH for a template, use the pipe-target debug in the CLI instead.

- `ned-cisco-ios-xr-pe1.trace`—If device trace is turned on, a trace file is created for each device. The file location is not configured in `wae.conf` but is configured when device trace is turned on, such as in the CLI.

Syslog

Using BSD or IETF syslog format (RFC5424), WAE can syslog to a local or remote syslog server. You can use the `wae.conf` file to choose which logs to save to syslog: `ncs.log`, `devel.log`, `netconf.log`, or `snmp.log`.

The following example shows a common syslog configuration:

```
<syslog-config>
  <facility>daemon</facility>

  <udp>
     <enabled>false</enabled>
     <host>127.0.0.1</host>
     <port>895</port>
  </udp>

  <syslog-servers>
    <server>
      <host>127.0.0.2</host>
      <version>1</version>
    </server>
    <server>
      <host>127.0.0.3</host>
      <port>7900</port>
      <facility>local4</facility>
      </server>
  </syslog-servers>
</syslog-config>

<ncs-log>
  <enabled>true</enabled>
  <file>
   <name>./logs/ncs.log</name>
   <enabled>true</enabled>
  </file>
  <syslog>
    <enabled>true</enabled>
  </syslog>
</ncs-log>
```

Syslog Messages and Formats

The following table lists WAE syslog messages and formats.

Symbol	Format String	Comment
DAEMON_DIED	"Daemon ~s died"	An external database daemon closed its control socket.
DAEMON_TIMEOUT	"Daemon ~s timed out"	An external database daemon did not respond to a query.
NO_CALLPOINT	"no registration found for callpoint ~s of type=~s"	ConfD tried to populate an XML tree, but no code had registered under the relevant callpoint.
CDB_DB_LOST	"CDB: lost DB, deleting old config"	CDB found its data schema file but not its data files. CDB recovered by starting from an empty database.
CDB_CONFIG_LOST	"CDB: lost config, deleting DB"	CDB found its data files but not its schema file. CDB recovered by starting from an empty database.

Symbol	Format String	Comment
CDB_UPGRADE_FAILED	"CDB: Upgrade failed: ~s"	Automatic CDB upgrade failed, meaning the data model was changed in a way that is not supported.
CDB_INIT_LOAD	"CDB load: processing file: ~s"	CDB is processing an initialization file.
CDB_OP_INIT	"CDB: Operational DB re-initialized"	The operational database was deleted and reinitialized because of an upgrade or a corrupt file.
CDB_CLIENT_TIMEOUT	"CDB client (~s) timed out, waiting for ~s"	A CDB client failed to answer within the timeout period and was disconnected.
INTERNAL_ERROR	"Internal error: ~s"	A ConfD internal error occurred and should be reported to Cisco technical support.
AAA_LOAD_FAIL	"Failed to load AAA: ~s"	Failed to load the AAA data because the external database is misbehaving or AAA is mounted or populated badly.
EXTAUTH_BAD_RET	"External auth program (user=~s) ret bad output: ~s"	Authentication is external and the external program returned badly formatted data.
BRIDGE_DIED	"confd_aaa_bridge died - ~s"	ConfD is configured to start the confd_aaa_bridge and the C program died.
PHASE0_STARTED	"ConfD phase0 started"	ConfD has started its start phase 0.
PHASE1_STARTED	"ConfD phase1 started"	ConfD has started its start phase 1.
STARTED	"ConfD started vsn: ~s"	ConfD has started.
UPGRADE_INIT_STARTED	"Upgrade init started"	In-service upgrade initialization started.
UPGRADE_INIT_SUCCEEDED	"Upgrade init succeeded"	In-service upgrade initialization succeeded.
UPGRADE_PERFORMED	"Upgrade performed"	In-service upgrade was performed but not yet committed.
UPGRADE_COMMITTED	"Upgrade committed"	In-service upgrade was committed.
UPGRADE_ABORTED	"Upgrade aborted"	In-service upgrade was terminated.
CONSULT_FILE	"Consulting daemon configuration file ~s"	ConfD is reading its configuration file.
STOPPING	"ConfD stopping (~s)"	ConfD is stopping (for example, due to **confd --stop**).
RELOAD	"Reloading daemon configuration"	Initiated daemon configuration reload.
BADCONFIG	"Bad configuration: ~s:~s: ~s"	confd.conf contains bad data.
WRITE_STATE_FILE_FAILED	"Writing state file failed: ~s: ~s (~s)"	Failed to write a state file.

Symbol	Format String	Comment
READ_STATE_FILE_FAILED	"Reading state file failed: ~s: ~s (~s)"	Failed to read a state file.
SSH_SUBSYS_ERR	"ssh protocol subsys - ~s"	Client did not send the **\"subsystem\"** command correctly.
SESSION_LIMIT	"Session limit of type '~s' reached, rejected new session request"	Session limit reached; new session request was rejected.
CONFIG_TRANSACTION_LIMIT	"Configuration transaction limit of type '~s' reached, rejected new transaction request"	Configuration transaction limit reached; new transaction request was rejected.
ABORT_CAND_COMMIT	"Aborting candidate commit, request from user, reverting configuration"	Terminating candidate commit due to user request. Reverting the configuration.
ABORT_CAND_COMMIT_TIMER	"Candidate commit timer expired, reverting configuration"	Candidate commit timer expired; reverting configuration.
ABORT_CAND_COMMIT_TERM	"Candidate commit session terminated, reverting configuration"	Candidate commit session terminated; reverting configuration.
ROLLBACK_REMOVE	"Found half created rollback0 file - removing and creating new"	Removing and recreating a rollback0 file that was found only half created.
ROLLBACK_REPAIR	"Found half created rollback0 file - repairing"	Repairing a rollback0 file that was found only half created.
ROLLBACK_FAIL_REPAIR	"Failed to repair rollback files"	Failed to repair a rollback file.
ROLLBACK_FAIL_CREATE	"Error while creating rollback file: ~s: ~s"	An error occurred while creating a rollback file.
ROLLBACK_FAIL_RENAME	"Failed to rename rollback file ~s to ~s: ~s"	Failed to rename a rollback file.
NS_LOAD_ERR	"Failed to process namespace ~s: ~s"	System failed to process a loaded namespace.
NS_LOAD_ERR2	"Failed to process namespaces: ~s"	System failed to process a loaded namespace.
FILE_LOAD_ERR	"Failed to load file ~s: ~s"	System failed to load a file in its load path.
FILE_LOADING	"Loading file ~s"	System is starting to load a file.
SKIP_FILE_LOADING	"Skipping file ~s: ~s"	System skipped a file.
FILE_LOAD	"Loaded file ~s"	System loaded a file.
LISTENER_INFO	"~s to listen for ~s on ~s:~s"	ConfD starts or stops to listen for incoming connections.
NETCONF_HDR_ERR	"Got bad NETCONF TCP header"	The clear text header that indicates users and groups was formatted badly.

Symbol	Format String	Comment
LIB_BAD_VSN	"Got library connect from wrong version (~s, expected ~s)"	An application connecting to ConfD used a library version that does not match the ConfD version (for example, an old version of the client library).
LIB_BAD_SIZES	"Got connect from library with insufficient keypath depth/keys support (~s/ ~s, needs ~s/~s)"	An application connecting to ConfD used a library version that cannot handle the depth and the number of keys used by the data model.
LIB_NO_ACCESS	"Got library connect with failed access check: ~s"	An access check failure occurred when an application connected to ConfD.
SNMP_NOT_A_TRAP	"SNMP gateway: Non-trap received from ~s"	A UDP package was received on the trap receiving port, but it's not an SNMP trap.
SNMP_TRAP_V1	"SNMP gateway: V1 trap received from ~s"	An SNMPv1 trap was received on the trap receiving port, but forwarding v1 traps is not supported.
SNMP_TRAP_NOT_FORWARDED	"SNMP gateway: Can't forward trap from ~s; ~s"	An SNMP trap was not forwarded.
SNMP_TRAP_UNKNOWN_ SENDER	"SNMP gateway: Not forwarding trap from ~s; the sender is not recognized"	An SNMP trap was supposed to be forwarded, but the sender was not listed in confd.conf.
SNMP_TRAP_OPEN_PORT	"SNMP gateway: Can't open trap listening port ~s: ~s"	Could not open the port for listening to SNMP traps.
SNMP_TRAP_NOT_RECOGNIZED	"SNMP gateway: Can't forward trap with OID ~s from ~s; There is no notification with this OID in the loaded models"	An SNMP trap was received on the trap receiving port, but its definition is unknown.
XPATH_EVAL_ERROR1	"XPath evaluation error: ~s for ~s"	An error occurred while evaluating an xpath expression.
XPATH_EVAL_ERROR2	"XPath evaluation error: '~s' resulted in ~s for ~s"	An error occurred while evaluating an xpath expression.
CANDIDATE_BAD_FILE_FORMAT	"Bad format found in candidate db file ~s; resetting candidate"	The candidate database file has a bad format. The candidate database is reset to an empty database.
CANDIDATE_CORRUPT_FILE	"Corrupt candidate db file ~s; resetting candidate"	The candidate database file is corrupt and cannot be read. The candidate database is reset to an empty database.
MISSING_DES3CBC_ SETTINGS	"DES3CBC keys were not found in confd.conf"	DES3CBC keys were not found in confd.conf.

Symbol	Format String	Comment
MISSING_AESCFB128_ SETTINGS	"AESCFB128 keys were not found in confd.conf"	AESCFB128 keys were not found in confd.conf.
SNMP_MIB_LOADING	"Loading MIB: ~s"	The SNMP agent is loading a MIB file.
SNMP_CANT_LOAD_MIB	"Can't load MIB file: ~s"	The SNMP agent failed to load a MIB file.
SNMP_WRITE_STATE_ FILE_FAILED	"Write state file failed: ~s: ~s"	Failed to write the SNMP agent state file.
SNMP_READ_STATE_ FILE_FAILED	"Read state file failed: ~s: ~s"	Failed to read the SNMP agent state file.
SNMP_REQUIRES_CDB	"Can't start SNMP. CDB is not enabled"	CDB must be enabled before the SNMP agent can start.
FXS_MISMATCH	"Fxs mismatch, slave is not allowed"	A secondary connected to a primary with different fxs files.
TOKEN_MISMATCH	"Token mismatch, slave is not allowed"	A secondary connected to a primary with a bad authorization token.
HA_SLAVE_KILLED	"Slave ~s killed due to no ticks"	A secondary node did not produce its ticks.
HA_DUPLICATE_NODEID	"Nodeid ~s already exists"	A secondary arrived with a node ID that already exists.
HA_FAILED_CONNECT	"Failed to connect to master: ~s"	An attempted library to become a secondary call failed because the secondary could not connect to the primary.
HA_BAD_VSN	"Incompatible HA version (~s, expected ~s), slave is not allowed"	A secondary connected to a primary with an incompatible HA protocol version.
NETCONF	"~s"	NETCONF traffic log message.
DEVEL_WEBUI	"~s"	Developer web UI log message.
DEVEL_AAA	"~s"	Developer AAA log message.
DEVEL_CAPI	"~s"	Developer C API log message.
DEVEL_CDB	"~s"	Developer CDB log message.
DEVEL_CONFD	"~s"	Developer ConfD log message.
DEVEL_SNMPGW	"~s"	Developer SNMP gateway log message.
DEVEL_SNMPA	"~s"	Developer SNMP agent log message.
NOTIFICATION_REPLAY_ STORE_FAILURE	"~_s"	A failure occurred in the built-in notification replay store.
EVENT_SOCKET_TIMEOUT	"Event notification subscriber with bitmask ~s timed out, waiting for ~s"	An event notification subscriber did not reply within the configured timeout period.

Symbol	Format String	Comment
EVENT_SOCKET_WRITE_BLOCK	"~s"	Write on an event socket was blocked for too long.
COMMIT_UN_SYNCED_DEV	"Committed data towards device ~s which is out of sync"	Data was committed toward a device with a bad or unknown sync state.
NCS_SNMP_INIT_ERR	"Failed to locate snmp_init.xml in loadpath ~s"	Failed to locate snmp_init.xml in the load path.
NCS_JAVA_VM_START	"Starting the NCS Java VM"	Starting the NCS Java VM.
NCS_JAVA_VM_FAIL	"The NCS Java VM ~s"	An NCS Java VM failure or timeout occurred.
NCS_PACKAGE_SYNTAX_ERROR	"Failed to load NCS package: ~s; syntax error in package file"	Syntax error in package file.
NCS_PACKAGE_DUPLICATE	"Failed to load duplicate NCS package ~s: (~s)"	Duplicate package found.
NCS_PACKAGE_COPYING	"Copying NCS package from ~s to ~s"	A package was copied from the load path to a private directory.
NCS_PACKAGE_UPGRADE_ABORTED	"NCS package upgrade failed with reason '~s'"	The CDB upgrade was terminated, implying that the CDB is untouched. However, the package state changed.
NCS_PACKAGE_BAD_NCS_VERSION	"Failed to load NCS package: ~s; requires NCS version ~s"	Bad NCS version for the package.
NCS_PACKAGE_BAD_DEPENDENCY	"Failed to load NCS package: ~s; required package ~s of version ~s is not present (found ~s)"	Bad NCS package dependency.
NCS_PACKAGE_CIRCULAR_DEPENDENCY	"Failed to load NCS package: ~s; circular dependency found"	Circular NCS package dependency.
CLI_CMD	"CLI '~s'"	User executed a CLI command.
CLI_DENIED	"CLI denied '~s'"	Due to permissions, a user was denied from executing a CLI command.
BAD_LOCAL_PASS	"Provided bad password"	A locally configured user provided a bad password.
NO_SUCH_LOCAL_USER	"no such local user"	A non existing local user tried to log in.
PAM_LOGIN_FAILED	"pam phase ~s failed to login through PAM: ~s"	A user failed to log in through PAM.
PAM_NO_LOGIN	"failed to login through PAM: ~s"	A user failed to log in through PAM.

Symbol	Format String	Comment
EXT_LOGIN	"Logged in over ~s using externalauth, member of groups: ~s~s"	An externally authenticated user logged in.
EXT_NO_LOGIN	"failed to login using externalauth: ~s"	External authentication failed for a user.
GROUP_ASSIGN	"assigned to groups: ~s"	A user was assigned to a set of groups.
GROUP_NO_ASSIGN	"Not assigned to any groups - all access is denied"	A user was logged in but was not assigned to any groups.
MAAPI_LOGOUT	"Logged out from maapi ctx=~s (~s)"	A management agent API (MAAPI) user was logged out.
SSH_LOGIN	"logged in over ssh from ~s with authmeth:~s"	A user logged into ConfD's built-in SSH server.
SSH_LOGOUT	"Logged out ssh <~s> user"	A user was logged out from ConfD's built-in SSH server.
SSH_NO_LOGIN	"Failed to login over ssh: ~s"	A user failed to log in to ConfD's built-in SSH server.
NOAAA_CLI_LOGIN	"logged in from the CLI with aaa disabled"	A user used the --noaaa flag to confd_cli.
WEB_LOGIN	"logged in through Web UI from ~s"	A user logged in through the web UI.
WEB_LOGOUT	"logged out from Web UI"	A web UI user logged out.
WEB_CMD	"WebUI cmd '~s'"	A user executed a web UI command.
WEB_ACTION	"WebUI action '~s'"	A user executed a web UI action.
WEB_COMMIT	"WebUI commit ~s"	A user performed a web UI commit.
SNMP_AUTHENTICATION_FAIL	"ESDNMP authentication failed: ~s"	An SNMP authentication failed.
LOGIN_REJECTED	"~s"	Authentication for a user was rejected by application callback.
COMMIT_INFO	"commit ~s"	Information about configuration changes committed to the running data store.
CLI_CMD_DONE	"CLI done"	CLI command finished successfully.
CLI_CMD_ABORTED	"CLI aborted"	CLI command terminated.
NCS_DEVICE_OUT_OF_SYNC	"NCS device-out-of-sync Device '~s' Info '~s'"	A check-sync action reported out-of-sync for a device.
NCS_SERVICE_OUT_OF_SYNC	"NCS service-out-ofsync Service '~s' Info '~s'"	A check-sync action reported out-of-sync for a service.
NCS_PYTHON_VM_START	"Starting the NCS Python VM"	Starting the NCS Python VM.

Symbol	Format String	Comment
NCS_PYTHON_VM_FAIL	"The NCS Python VM ~s"	The NCS Python VM failed or timed out.
NCS_SET_PLATFORM_ DATA_ERRORS	"NCS Device '~s' failed to set platform data Info '~s'"	The device failed to set the platform operational data at connect.
NCS_SMART_LICENSING_ START	"Starting the NCS Smart Licensing Java VM"	Starting the NCS Smart Licensing Java VM.
NCS_SMART_LICENSING_FAIL	"The NCS Smart Licensing Java VM ~s"	The NCS Smart Licensing Java VM failed or timed out.
NCS_SMART_LICENSING_ GLOBAL_NOTIFICATION	"Smart Licensing Global Notification: ~s"	Smart Licensing global notification.
NCS_SMART_LICENSING_ ENTITLEMENT_NOTIFICATION	"Smart Licensing Entitlement Notification: ~s"	Smart Licensing entitlement notification.
NCS_SMART_LICENSING_ EVALUATION_COUNTDOWN	"Smart Licensing evaluation time remaining: ~s"	Smart Licensing evaluation time remaining.
DEVEL_SLS	"~s"	Developer Smart Licensing API log message.
JSONRPC_REQUEST	"JSON-RPC: '~s' with JSON params ~s"	JSON-RPC method requested.
DEVEL_ECONFD	"~s"	Developer econfd API log message.
CDB_FATAL_ERROR	"fatal error in CDB: ~s"	CDB encountered an unrecoverable error.
LOGGING_STARTED	"Daemon logging started"	Logging subsystem started.
LOGGING_SHUTDOWN	"Daemon logging terminating, reason: ~s"	Logging subsystem terminated.
REOPEN_LOGS	"Logging subsystem, reopening log files"	Logging subsystem reopened log files.
OPEN_LOGFILE	"Logging subsystem, opening log file '~s' for ~s"	Indicate target file for certain type of logging.
LOGGING_STARTED_TO	"Writing ~s log to ~s"	Write logs for a subsystem to a specific file.
LOGGING_DEST_CHANGED	"Changing destination of ~s log to ~s"	The target log file will change to another file.
LOGGING_STATUS_CHANGED	"~s ~s log"	Notify a change of logging status (enabled/disabled) for a subsystem.
ERRLOG_SIZE_CHANGED	"Changing size of error log (~s) to ~s (was ~s)"	Notify a change of log size for an error log.
CGI_REQUEST	"CGI: '~s' script with method ~s"	CGI script requested.
MMAP_SCHEMA_FAIL	"Failed to setup the shared memory schema"	Failed to set up the shared memory schema.

Symbol	Format String	Comment
KICKER_MISSING_SCHEMA	"Failed to load kicker schema"	Failed to load the kicker schema.
JSONRPC_REQUEST_ IDLE_TIMEOUT	"Stopping session due to idle timeout: ~s"	JSON-RPC idle timeout.
JSONRPC_REQUEST_ ABSOLUTE_TIMEOUT	"Stopping session due to absolute timeout: ~s"	JSON-RPC absolute timeout.

Database Locking

This section explains the different locks that exist in WAE and how they interact.

Global Locks

The WAE management backplane keeps a lock on the data store: *running*. This lock is known as the global lock and provides a mechanism to grant exclusive access to the data store. The global lock is the only lock that can explicitly be taken through a northbound agent—for example, by the NETCONF <lock> operation—or by calling Maapi.lock().

A global lock can be taken for the entire data store, or it can be a partial lock (for a subset of the data model). Partial locks are exposed through NETCONF and MAAPI.

An agent can request a global lock to ensure that it has exclusive write access. When an agent holds a global lock, no one else can write to that data store. This behavior is enforced by the transaction engine. A global lock on running is granted to an agent if there are no other lock holders (including partial locks), and if all data providers approve the lock request. Each data provider (CDB or external data provider) has its lock() callback invoked to refuse or accept the lock. The output of **ncs --status** includes the lock status.

Transaction Locks

A northbound agent starts a user session towards the WAE management backplane. Each user session can then start multiple transactions. A transaction is either read/write or read-only.

The transaction engine has its internal locks toward the running data store. These transaction locks exist to serialize configuration updates toward the data store and are separate from global locks.

When a northbound agent wants to update the running data store with a new configuration, it implicitly grabs and releases the transactional lock. The transaction engine manages the lock as it moves through the transaction state machine. No API exposes the transactional lock to the northbound agent.

When the transaction engine wants to take a lock for a transaction (for example, when entering the validate state), it first checks that no other transaction has the lock. It then checks that no user session has a global lock on that data store. Finally, it invokes each data provider with a transLock() callback.

Northbound Agents and Global Locks

In contrast to implicit transactional locks, some northbound agents expose explicit access to global locks. The management API exposes global locks by providing Maapi.lock() and Maapi.unlock() methods (and the

corresponding Maapi.lockPartial() Maapi.unlockPartial() for partial locking). Once a user session is established (or attached to), these functions can be called.

In the CLI, global locks are taken when entering different configure modes, as follows:

- **config exclusive**—Takes the running data store global lock.

- **config terminal**—Does not grab any locks.

The CLI keeps the global lock until the configure mode is exited.

The Expert Mode behaves in the same way as the CLI: it has edit tabs called **Edit private** and **Edit exclusive**, which correspond to the CLI modes described above.

The NETCONF agent translates the <lock> operation into a request for a global lock for the requested data store. Partial locks are also exposed through the partial-lock rpc.

External Data Providers and CDB

An external data provider is not required to implement the lock() and unlock() callbacks. WAE never tries to initiate the transLock() state transition toward a data provider while a global lock is taken. The reason for a data provider to implement the locking callbacks is if someone else can write to the data provider's database.

CDB ignores the lock() and unlock() callbacks (because the data provider interface is the only write interface towards it).

CDB has its own internal locks on the database. The running data store has a single write lock and multiple read locks. It is not possible to grab the write lock on a data store while there are active read locks on it. The locks in CDB exist to ensure that a reader always gets a consistent view of the data. (Confusion occurs if another user deletes configuration nodes in between calls to getNext() on YANG list entries.)

During a transaction transLock() takes a CDB read lock toward the transaction's data store and writeStart() tries to release the read lock and grab the write lock instead. A CDB external reader client implicitly takes a CDB read lock between Cdb.startSession() and Cdb.endSession(). This means that while a CDB client is reading, a transaction cannot pass through writeStart(). Conversely, a CDB reader cannot start while a transaction is in between writeStart() and commit() or abort().

The operational store in CDB does not have any locks; WAE's transaction engine can only read from it. CDB client writes are atomic per write operation.

Lock Impact on User Sessions

When a session tries to modify a data store that is locked, it fails. For example, the CLI might print:

```
admin@wae(config)# commit
Aborted: the configuration database is locked
```

Because some locks are short-lived (such as a CDB read lock), WAE is configured by default to retry the failed operation for a configurable length of time. If the data store remains locked after this time, the operation fails.

To configure the retry timeout, set the /ncs-config/commit-retry-timeout value in wae.conf.

Security

WAE requires privileges to perform certain tasks. Depending on the target system, the following tasks might require root privileges:

- Binding to privileged ports. The `wae.conf` configuration file specifies which port numbers WAE should bind(2) to. If a port number is lower than 1024, WAE usually requires root privileges unless the target operating system allows WAE to bind to these ports as a non-root user.

- If PAM is used for authentication, the program installed as `$NCS_DIR/lib/ncs/priv/pam/epam` acts as a PAM client. Depending on the local PAM configuration, this program might require root privileges. If PAM is configured to read the local `passwd` file, the program must either run as root, or be setuid root. If the local PAM configuration instructs WAE to run for example pam_radius_auth, root privileges might not be required, depending on the local PAM installation.

- If the CLI is used to create CLI commands that run executables, modify the permissions of the `$NCS_DIR/lib/ncs/priv/ncs/cmdptywrapper` program.

To run an executable as root or as a specific user, make cmdptywrapper setuid root:

```
# chown root cmdptywrapper
# chmod u+s cmdptywrapper
```

Failing that, all programs are executed as the user running the WAE daemon. If that user is root, you need not perform the chmod operations above.

Failing that, all programs are executed as the user running the confd daemon. If that user is root, you need not perform the preceding chmod operations.

For executables that run via actions, modify the permissions of the `$NCS_DIR/lib/ncs/priv/ncs/cmdwrapper` program:

```
# chown root cmdwrapper
# chmod u+s cmdwrapper
```

WAE can be instructed to terminate NETCONF over clear text TCP, which is useful for debugging (NETCONF traffic can be captured and analyzed) and when providing a local proprietary transport mechanism other than SSH. Clear text TCP termination is not authenticated; the clear text client simply tells WAE which user the session should run as. The assumption is that authentication is already done by an external entity, such as an SSH server. If clear text TCP is enabled, WAE must bind to localhost (127.0.0.1) for these connections.

Client libraries connect to WAE. For example, the CDB API is TCP-based and a CDB client connects to WAE. WAE learns which address to use for these connections through the `wae.conf` parameters `/ncs-config/ncs-ipc-address/ip` (the default address is 127.0.0.1) and `/ncs-config/ncs-ipcaddress/port` (the default port is 4565).

WAE multiplexes different kinds of connections on the same socket (IP and port combination). The following programs connect on the socket:

- Remote commands, such as **ncs --reload**.

- CDB clients.

- External database API clients.

- Management agent API (MAAPI) clients.

- The ncs_cli program.

By default, the preceding programs are considered trusted. MAAPI clients and the ncs_cli authenticate users before connecting to WAE. CDB clients and external database API clients are considered trusted and do not have to authenticate.

Because the ncs-ipc-address socket allows full, unauthenticated access to the system, it is important to ensure that the socket is not accessible from untrusted networks. You can also restrict access to the ncs-ipc-address socket by means of an access check. See .

Restrict Access to the IPC Port

By default, clients connecting to the IPC port are considered trusted; no authentication is required. To prevent remote access, WAE relies on the use of 127.0.0.1 for `/ncs-config/ncs-ipc-address/ip`. However, you can restrict access to the IPC port by configuring an access check.

To enable the access check, set the `wae.conf` element `/ncs-config/ncs-ipc-accesscheck/enabled` to **true**, and specify a filename for `/ncs-config/ncs-ipc-accesscheck/filename`. The file should contain a shared secret (a random-character string). Clients connecting to the IPC port must provide a challenge handshake before they are granted access to WAE functions.

> **Note** The access permissions on this file must be restricted via OS file permissions, such that the file can only be read by the WAE daemon and client processes that are allowed to connect to the IPC port. For example, if both the daemon and the clients run as root, the file can be owned by root and have only "read by owner" permission (mode 0400). Another possibility is to create a group that only the daemon and the clients belong to, set the group ID of the file to that group, and have only "read by group" permission (mode 040).

To provide the secret to the client libraries and instruct them to use the access check handshake, set the environment variable NCS_IPC_ACCESS_FILE to the full path name of the file that contains the secret. This is sufficient for all clients mentioned above; there is no need to change the application code to enable this check.

> **Note** The access check must be either enabled or disabled for both the daemon and the clients. For example, if the `wae.conf` element `/ncsconfig/ncs-ipc-access-check/enabled` is not set to **true**, but clients are started with the environment variable NCS_IPC_ACCESS_FILE pointing to a file with a secret, the client connections fail.

Back Up and Restore the WAE Configuration

With the YANG run-time framework, you can easily back up and restore the WAE configuration. We recommend that you back up the WAE configuration before starting any collection (that is, before any operational data is populated).

• To back up a WAE configuration:

```
admin@wae% save /home/wae/wae-backup.cfg
```

The preceding command backs up both the configuration data and the operational data.

• To restore a WAE configuration:

```
[wae@wae ~]$ ncs_load -l -m -F j wae-backup.cfg
```

WAE Diagnostics

Cisco WAE includes a diagnostics utility that can:

• run diagnostic checks on the health of the system and make recommendations.

• collect all required data/health check reports that might be required for engineers to troubleshoot the issue.

The tool has an extensible framework where additional health-check scripts can be added. Additional health-check scripts can either be in python/shell and must be placed under `<wae-install-directory>/bin/diagnostics/` directory.

WAE Diagnostic Tool usage

WAE diagnostics can be executed using the `wae-diagnostics` commmand:

✎

Note   Source `waerc` before running the command.

```
wae-diagnostics [-h] [-c] [-D] [-j] [-e] [-d] [--run-diagnostics] [-o OUT_DIR] -r RUN_DIR
-i INSTALL_DIR
```

where

Required arguments	
-r RUN_DIR	run-dir--RUN_DIR run directory path
-i INSTALL_DIR	install-dir--INSTALL_DIR install directory path
Optional arguments	
-h	help--Show this help message and exit
-c	collect-logs--Collects and collates all logs and troubleshooting data. Excludes DB files unless specified using --collect-db-files
-D	collect-db-files--Collects db files

-j	java-stats--Collects java thread stats
-e	enable-debug--Sets logs levels to debug
-d	disable-debug--Disables debug log level
--run-diagnostics	runs diagnostics on WAE
-o OUT_DIR	out-dir--OUT_DIR output directory path. The data archive will be created in this directory, if not specified, the data archive will be created in current directory.

Sample:

• The following command runs diagnostic checks on the WAE installation, collect diagnostics information including logs.

```
wae-diagnostics -r <run-directory-path> -i <install-dir-path>
```

• The following command runs diagnostic checks on the WAE installation, collect diagnostics information including logs, network data and JVM data.

```
wae-diagnostics -cDj -r <run-directory-path> -i <install-dir-path>
```

Security

- Core Security Concepts, on page 193

Core Security Concepts

If you are an administrator and are looking to optimize the security of your product, you should have a good understanding of the following security concepts.

HTTPS

Hypertext Transfer Protocol Secure (HTTPS) uses Secure Sockets Layer (SSL) or its subsequent standardization, Transport Layer Security (TLS), to encrypt the data transmitted over a channel. Several vulnerabilities have been found in SSL, so now supports TLS only.

> **Note** TLS is loosely referred to as SSL often, so we will also follow this convention.

SSL employs a mix of privacy, authentication, and data integrity to secure the transmission of data between a client and a server. To enable these security mechanisms, SSL relies upon certificates, private-public key exchange pairs, and Diffie-Hellman key agreement parameters.

SSL Certificates

SSL certificates and private-public key pairs are a form of digital identification for user authentication and the verification of a communication partner's identity. Certificate Authorities (CAs), such as VeriSign and Thawte, issue certificates to identify an entity (either a server or a client). A client or server certificate includes the name of the issuing authority and digital signature, the serial number, the name of the client or server that the certificate was issued for, the public key, and the certificate's expiration date. A CA uses one or more signing certificates to create SSL certificates. Each signing certificate has a matching private key that is used to create the CA signature. The CA makes signed certificates (with the public key embedded) readily available, enabling anyone to use them to verify that an SSL certificate was actually signed by a specific CA.

In general, setting up certificates involve the following steps:

1. Generating an identity certificate for a server.

2. Installing the identity certificate on the server.

3. Installing the corresponding root certificate on your client or browser.

The specific tasks you need to complete will vary, depending on your environment.

1-Way SSL Authentication

This authentication method is used when a client needs assurance that it is connecting to the right server (and not an intermediary server), making it suitable for public resources like online banking websites. Authentication begins when a client requests access to a resource on a server. The server on which the resource resides then sends its server certificate (also known as an SSL certificate) to the client in order to verify its identity. The client then verifies the server certificate against another trusted object: a server root certificate, which must be installed on the client or browser. After the server has been verified, an encrypted (and therefore secure) communication channel is established. At this point, the server prompts for the entry of a valid username and password in an HTML form. Entering user credentials after an SSL connection is established protects them from being intercepted by an unauthorized party. Finally, after the username and password have been accepted, access is granted to the resource residing on the server.

Note A client might need to store multiple server certificates to enable interaction with multiple servers.

To determine whether you need to install a root certificate on your client, look for a lock icon in your browser's URL field. If you see this icon, this generally indicates that the necessary root certificate has already been installed. This is usually the case for server certificates signed by one of the bigger Certifying Authorities (CAs), because root certificates from these CAs are included with popular browsers.

If your client does not recognize the CA that signed a server certificate, it will indicate that the connection is not secure. This is not necessarily a bad thing. It just indicates that the identity of the server you want to connect has not been verified. At this point, you can do one of two things: First, you can install the necessary root certificate on your client or browser. A lock icon in your browser's URL field will indicate the certificate was installed successfully. And second, you can install a self-signed certificate on your client. Unlike a root certificate, which is signed by a trusted CA, a self-signed certificate is signed by the person or entity that created it. While you can use a self-signed certificate to create an encrypted channel, understand that it carries an inherent amount of risk because the identity of the server you are connected with has not been verified.

APPENDIX A

Additional WAE CLI Commands

This section contains the following topics:

Commit Flags

Commit flags modify transaction semantics. Use a commit flag when issuing a **commit** command:

commit *<flag>*

The following table lists some of commonly used flags.

Command	Description
and-quit	Exits to CLI operational mode after a commit.
bypass-commit-queue	Attempts to commit directly, bypassing the commit queue. This flag is relevant only when the commit queue is used by default (by the configuration item `/devices/global-settings/commit-queue/enabled-bydefault`). The operation fails if the commit queue contains entries that affect the same device(s) as the transaction to be committed.
check	Validates the pending configuration changes. Equivalent to the **validate** command.
comment | **label**	Adds a commit comment or label the is visible in compliance reports, rollback files, and so on.
dry-run	Validates and displays the configuration changes, but does not perform the actual commit. Neither CDB nor devices are affected. Various output formats are supported.
no-networking	Validates the configuration changes and updates the CDB, but does not update the actual devices. This is equivalent to first setting the admin state-state to southbound locked, then issuing a standard commit. In both cases the configuration changes are not committed to actual devices. If the commit implies changes, it makes the device out-of-sync.

`no-out-of-sync-check`	Commits even if the device is out-of-sync. This flag can be used in scenarios where you know the change is not in conflict with what is on the device, and you don't want to perform a **sync-from** first. Use **device compare-config** to verify the result. If the commit implies changes, it makes the device out-of-sync.
`no-revision-drop`	Fails if devices have obsolete device models. When WAE connects to a NETCONF device, the version of the device data model is discovered. Different devices in the network might have different versions. When WAE sends configurations to devices, by default it drops any configuration that only exists in later models than the device supports.
`through-commit-queue`	Although the configuration change is committed to CDB immediately, it is not committed to the actual device. Instead, to increase transaction throughput, the config change is queued for eventual commit. This enables the use of the commit queue feature for individual commit commands without enabling it by default.

All WAE command can have pipe commands. For example, the **details** pipe command provides feedback on the steps performed in the commit:

```
wae% commit | details
```

To enable debugging on all templates, use the **debug** pipe command:

```
wae% commit | debug template
```

If you use many templates during configuration, the debug output can be overwhelming. You can limit debug information to just one template, as shown in the following example for a template named *l3vpn*:

```
wae% commit | debug template l3vpn
```

Device Actions

Actions for devices can be performed globally on the `/devices` path, and for individual devices on `/devices/device/name`. Many actions are also available on device groups and device ranges.

The following table lists device actions.

Command	Description
`check-sync`	Checks if the WAE copy of the device configuration is in sync with the actual device configuration. This operation compares only a signature of the configuration from the device; it does not compare the entire configuration. The signature is implemented as a transaction-id, time-stamp, or hash-sum. The corresponding NED must support the capability. If the output says unsupported, you must use a full **device compare-config** command.
`check-yang-modules`	Checks if WAE and the devices have compatible YANG modules.
`clear-trace`	Clears all trace files.

commit-queues	Displays a list of queued commits.
connect	Sets up sessions to unlocked devices. This action is not used in real operational scenarios, because WAE automatically establishes connections on demand. However, this action is useful for test purposes when installing new NEDs, adding devices, and so on.
disconnect	Closes the session to the device.
sync-from	Synchronizes the WAE copy of the device configuration by reading the actual device configuration. The change is committed immediately to WAE and cannot be rolled back. If any service created a configuration on the device, the corresponding service might be out of sync. To reconcile this discrepancy, use the commands **service check-sync** and **service re-deploy**.
sync-to	Synchronizes the device configuration by pushing the WAE copy to the devices. (This action cannot be rolled back.)

Service Actions

Many of the preceding device operations can be combined with the option **no-networking**, which performs all updates only in the configuration database and makes the devices out of sync. The updates can be pushed to the network later. (This action is the same as setting the devices in admin-state southbound-locked.)

The following table lists service actions.

Command	Description
check-sync	Verifies that the service and the associated device configuration is in sync. Any differences are displayed in a chosen out-format. If configuration changes were made out-of-band, a **deep-check-sync** is required to detect an out-of-sync condition.
deep-check-sync	Validates whether the actual devices are configured according to the service. Use **re-deploy** to reconcile the service.
get-modifications	Gets the configuration data created by the service.
re-deploy	Reruns the service logic—taking into account all service data—and generates a diff using the device configuration in the configuration database. Sends the configuration diff to the devices. This action is useful when: • A **device sync-from** action has been performed to incorporate an out-of-band change. • Data referenced by the service—topology information, QoS policy definitions, and so on—has changed. This action is idempotent. If no configuration diff exists, nothing needs to be done. The WAE general principle of minimum change applies.
un-deploy	Undoes the effects of the service on the network. This action removes the configuration from the actual devices and from the WAE configuration database.

wae.conf Configuration Parameters

The following table lists the `wae.conf` configuration parameters and their type (in parentheses) and default values (in brackets). Parameters are written using a path notation to make it easier to see how they relate to each other.

Parameter	Description
`/ncs-config`	WAE configuration.
`/ncs-config/db-mode (running) [running]`	This feature is deprecated; WAE supports only running db-mode. It is not a requirement to set this leaf; it is retained only for backward compatibility.
`/ncs-config/ncs-ipc-address`	WAE listens by default on 127.0.0.1:4569 for incoming TCP connections from WAE client libraries, such as CDB, MAAPI, the CLI, the external database API, as well as commands from the ncs script (such as 'ncs --reload'). The IP address and port can be changed. If they are changed, all clients using MAAPI, CDB, and so on must be recompiled to handle this. **Caution** There are severe security implications involved if WAE is instructed to bind(2) to anything but localhost. Use the IP 0.0.0.0 if you want WAE to listen(2) on all IPv4 addresses.
`/ncs-config/ncs-ipc-address/ip (ipv4-address | ipv6-address) [127.0.0.1]`	The IP address that WAE listens on for incoming connections from the Java library.
`/ncs-config/ncs-ipc-address/port (port-number) [4569]`	The port number that WAE listens on for incoming connections from the Java library.
`/ncs-config/ncs-ipc-extra-listen-ip (ipv4-address | ipv6-address)`	This parameter can be given multiple times. It lists additional IPs to which to bind the WAE IPC listener. This is useful if you don't want to use the wildcard 0.0.0.0 address in order to never expose the WAE IPC to certain interfaces.
`/ncs-config/ncs-ipc-access-check`	WAE can be configured to restrict access for incoming connections to the IPC listener sockets. The access check requires that connecting clients prove possession of a shared secret.
`/ncs-config/ncs-ipc-access-check/enabled (boolean) [false]`	If 'true', the access check for IPC connections is enabled.
`/ncs-config/ncs-ipc-access-check/filename (string)`	This parameter is mandatory. *filename* is the full path to a file containing the shared secret for the IPC access check. The file should be protected via OS file permissions, such that it can only be read by the WAE daemon and client processes that are allowed to connect to the IPC listener sockets.
`/ncs-config/enable-shared-memory-schema (boolean) [true]`	*enabled* is either true or false. If true, a C program starts and loads the schema into shared memory (which can then be accessed by Python, for example).
`/ncs-config/load-path`	—

Parameter	Description
`/ncs-config/load-path/dir (string)`	This parameter can be given multiple times. The *load-path* element contains any number of *dir* elements. Each *dir* element points to a directory path on disk that is searched for compiled and imported YANG files (.fxs files) and compiled clispec files (.ccl files) during daemon startup. WAE also searches the load path for packages at initial startup, or when requested by the /packages/reload action.
`/ncs-config/state-dir (string)`	This parameter is mandatory. This is where WAE writes persistent state data. It stores a private copy of all packages found in the load path, in a directory tree rooted at 'packages-in-use.cur' (also referenced by a symlink 'packages-in-use'). It is also used for the state file 'running.invalid', which exists only if the running database status is invalid, which occurs if one of the database implementations fails during the two-phase commit protocol. It is also used for 'global.data', which is used to store data that needs to be retained across reboots.
`/ncs-config/commit-retry-timeout (xs:duration | infinity) [infinity]`	Commit timeout in the WAE back plane. This timeout controls how long the commit operation in the CLI and the JSON-RPC API try to complete the operation when another entity is locking the database; for example, when another commit is in progress or when a managed object is locking the database.
`/ncs-config/max-validation-errors (uint32 | unbounded) [1]`	Controls how many validation errors are collected and presented to the user at a time.
`/ncs-config/notifications`	Defines NETCONF northbound notification settings.
`/ncs-config/notifications/event-streams`	Lists all available notification event streams.
`/ncs-config/notifications/event-streams/ stream`	Parameters for a single notification event stream.
`/ncs-config/notifications/event-streams/ stream/name (string)`	The name attached to a specific event stream.
`/ncs-config/notifications/event-streams/ stream/description (string)`	This parameter is mandatory. Descriptive text attached to a specific event stream.
`/ncs-config/notifications/event-streams/ stream/replay-support (boolean)`	This parameter is mandatory. Signals if replay support is available for a specific event stream.
`/ncs-config/notifications/event-streams/ stream/builtin-replay-store`	Parameters for the built-in replay store for this event stream. If replay support is enabled, WAE automatically stores all notifications on disk, ready to be replayed if a NETCONF manager asks for logged notifications. The replay store uses a set of wrapping log files on disk (of a certain number and size) to store the notifications. To achieve fast replay of notifications in a certain time range, the max size of each wrap log file should not be too large. If possible, use a larger number of wrap log files instead. If in doubt, use the recommended settings (see below).

Parameter	Description
`/ncs-config/notifications/event-streams/stream/builtin-replay-store/enabled (boolean) [false]`	If 'false', the application must implement its own replay support.
`/ncs-config/notifications/event-streams/stream/builtin-replay-store/dir (string)`	This parameter is mandatory. The disk location for the wrapping log files.
`/ncs-config/notifications/event-streams/stream/builtin-replay-store/max-size (tailf:size)`	This parameter is mandatory. The max size of each log wrap file. The recommended setting is approximately S10M.
`/ncs-config/notifications/event-streams/stream/builtin-replay-store/max-files (int64)`	This parameter is mandatory. The max number of log wrap files. The recommended setting is around 50 files.
`/ncs-config/opcache`	Controls the behavior of the operational data cache.
`/ncs-config/opcache/enabled (boolean) [false]`	If 'true', the cache is enabled.
`/ncs-config/opcache/timeout (uint64)`	This parameter is mandatory. The amount of time to keep data in the cache, in seconds.
`/ncs-config/hide-group`	Lists any hide groups that can be unhidden. There can be zero, one, or many hide-group entries in the configuration. If a hide group does not have a hide-group entry, it cannot be unhidden using the CLI 'unhide' command. However, it is possible to add a hide-group entry to the ncs.conf file and then use **ncs -- reload** to make it available in the CLI. This can be useful to enable, for example, a diagnostics hide group that you do not want accessible even using a password.
`/ncs-config/hide-group/name (string)`	Name of the hide group, which should correspond to a hide group name defined in a YANG module with 'tailf:hidden'.
`/ncs-config/hide-group/ password (tailf:md5-digest-string) []`	A password can optionally be specified for a hide group. If no password or callback is given, the hide group can be unhidden without giving a password. If a password is specified, the hide group cannot be enabled unless the password is entered. To completely disable a hide group (that is, make it impossible to unhide it), remove the entire hide-group container for that hide group.
`/ncs-config/hide-group/ callback (string)`	A callback can optionally be specified for a hide group. If no callback or password is given, the hide group can be unhidden without giving a password. If a callback is specified, the hide group cannot be enabled unless a password is entered and verified. The callback receives the name of the hide group, the name of the user issuing the unhide command, and the password. Callbacks make it possible to have short-lived unhide passwords and per-user unhide passwords.

Parameter	Description
`/ncs-config/cdb`	—
`/ncs-config/cdb/db-dir (string)`	*db-dir* is the directory on disk that CDB uses for its storage and any temporary files. It is also the directory where CDB searches for initialization files.
`/ncs-config/cdb/init-path`	—
`/ncs-config/cdb/init-path/dir (string)`	This parameter can be given multiple times. The *init-path* can contain any number of *dir* elements. Each *dir* element points to a directory path that CDB searches for .xml files before looking in *db-dir*. The directories are searched in the order in which they are listed.
`/ncs-config/cdb/client-timeout (xs:duration | infinity) [infinity]`	Specifies how long CDB waits for a response before considering a client unresponsive. If a client fails to call Cdb.syncSubscriptionSocket() within the timeout period, CDB logs a syslog of this failure and then, considering the client dead, closes the socket and proceeds with the subscription notifications. If set to infinity, CDB never times out waiting for a response from a client.
`/ncs-config/cdb/subscription-replay`	—
`/ncs-config/cdb/subscription-replay/enabled (boolean) [false]`	If enabled, it is possible to request a replay of the previous subscription notification to a new CDB subscriber.
`/ncs-config/cdb/replication (async | sync) [sync]`	When CDB replication is enabled (which it is when high-availability mode is enabled; see /ncs-config/ha), the CDB configuration stores can be replicated asynchronously or synchronously. With asynchronous replication, a transaction updating the configuration is allowed to complete as soon as the updates are sent to the connected secondary nodes. With the default synchronous replication, the transaction is suspended until the updates are completely propagated to the secondary nodes, and the subscribers on the secondary nodes (if any) have acknowledged their subscription notifications.
`/ncs-config/cdb/journal-compaction (automatic | manual) [automatic]`	Controls the way the CDB configuration store does its journal compaction. Never set to anything but the default 'automatic' unless there is an external mechanism that controls the compaction using the cdb_initiate_journal_compaction() API call.
`/ncs-config/cdb/operational`	Operational data can either be implemented by external callbacks, or stored in CDB (or a combination of both). The operational data store is used when data is to be stored in CDB.
`/ncs-config/cdb/operational/ db-dir (string)`	*db-dir* is the directory on disk that CDB operational uses for its storage and any temporary files. If left unset (default), the same directory as *db-dir* for CDB is used.
`/ncs-config/encrypted-strings`	*encrypted-strings* defines keys used to encrypt strings that adhere to the types tailf:des3-cbc-encryptedstring and tailf:aes-cfb-128-encrypted-string.
`/ncs-config/encrypted-strings/DES3CBC`	With DES3CBC, three 64-bit (8-byte) keys and a random initial vector are used to encrypt the string. The initVector leaf is only used when upgrading from earlier versions, but is retained for backward compatibility.

Parameter	Description
`/ncs-config/encrypted-strings/` `DES3CBC/key1 (hex8-value-type)`	This parameter is mandatory.
`/ncs-config/encrypted-strings/` `DES3CBC/key2 (hex8-value-type)`	This parameter is mandatory.
`/ncs-config/encrypted-strings/` `DES3CBC/key3 (hex8-value-type)`	This parameter is mandatory.
`/ncs-config/encrypted-strings/` `DES3CBC/initVector` `(hex8-value-type)`	—
`/ncs-config/encrypted-strings/` `AESCFB128`	With AESCFB128, one 128-bit (16-byte) key and a random initial vector are used to encrypt the string. The initVector leaf is only used when upgrading from earlier versions, but is retained for backward compatibility.
`/ncs-config/encrypted-strings/` `AESCFB128/key (hex16-value-type)`	This parameter is mandatory.
`/ncs-config/encrypted-strings/` `AESCFB128/initVector` `(hex16-value-type)`	—
`/ncs-config/crypt-hash`	*crypt-hash* specifies how clear-text values should be hashed for leafs of the types ianach:crypt-hash, tailf:sha-256-digest-string, and tailf:sha-512-digest-string.
`/ncs-config/crypt-hash/algorithm` `(md5 | sha-256 | sha-512) [md5]`	*algorithm* can be set to one of the values 'md5', 'sha-256', or 'sha-512', to choose the corresponding hash algorithm for hashing of clear-text input for the ianach:crypt-hash type.
`/ncs-config/crypt-hash/rounds` `(crypt-hash-rounds-type) [5000]`	For the 'sha-256' and 'sha-512' algorithms for the ianach:crypt-hash type, and for the tailf:sha-256-digest-string and tailf:sha-512-digest-string types, *rounds* specifies how many times the hashing loop should be executed. If a value other than the default 5000 is specified, the hashed format has 'rounds=N$', where N is the specified value, prepended to the salt. This parameter is ignored for the 'md5' algorithm for ianach:crypt-hash.
`/ncs-config/logs`	—
`/ncs-config/logs/syslog-config`	Shared settings for how to log to syslog. Logs can be configured to log to file or syslog. If a log is configured to log to syslog, the settings under /ncs-config/logs/syslog-config are used.
`/ncs-config/logs/syslog-config/version` `(bsd | 1) [bsd]`	*version* is either 'bsd' (traditional syslog) or '1' (new IETF syslog format: RFC 5424). '1' implies that /ncs-config/logs/syslog-config/udp/enabled must be set to true.

Parameter	Description
`/ncs-config/logs/syslog-config/facility (daemon | authpriv | local0 | local1 | local2 | local3 | local4 | local5 | local6 | local7 | uint32) [daemon]`	This facility setting is the default facility. It is also possible to set individual facilities in the different logs.
`/ncs-config/logs/syslog-config/udp`	—
`/ncs-config/logs/syslog-config/ udp/enabled (boolean) [false]`	If 'false', messages are sent to the local syslog daemon.
`/ncs-config/logs/syslog-config/ udp/host (string | ipv4-address | ipv6-address)`	This parameter is mandatory. *host* is either a domain name or an IPv4/IPv6 network address. UDP syslog messages are sent to this host.
`/ncs-config/logs/syslog-config/ udp/port (port-number) [514]`	*port* is a valid port number to be used in combination with /ncs-config/logs/syslog-config/udp/host.
`/ncs-config/logs/syslog-config/ syslog-servers`	This is an alternative way of specifying UDP syslog servers. If you configure the /ncs-config/logs/syslog-config/udp container, any configuration in this container is ignored.
`/ncs-config/logs/syslog-config/ syslog-servers/server`	A set of syslog servers that get a copy of all syslog messages.
`/ncs-config/logs/syslog-config/ syslog-servers/server/host (string | ipv4-address | ipv6-address)`	*host* is either a domain name or an IPv4/IPv6 network address. UDP syslog messages are sent to this host.
`/ncs-config/logs/syslog-config/ syslog-servers/server/port (port-number) [514]`	*port* is the UDP port number where this syslog server is listening.
`/ncs-config/logs/syslog-config/ syslog-servers/server/version (bsd | 1) [bsd]`	*version* is either 'bsd' (traditional syslog) or '1' (new IETF syslog format: RFC 5424).
`/ncs-config/logs/syslog-config/ syslog-servers/server/facility (daemon | authpriv | local0 | local1 | local2 | local3 | local4 | local5 | local6 | local7 | uint32) [daemon]`	—
`/ncs-config/logs/syslog-config/ syslog-servers/server/enabled (boolean) [true]`	If 'false', this syslog server does not get any UDP messages.
`/ncs-config/logs/ncs-log`	ncs-log is WAE's daemon log. Check this log for startup problems of the WAE daemon itself. This log is not rotated; use logrotate(8).

Parameter	Description
`/ncs-config/logs/ncs-log/ enabled (boolean) [true]`	If 'true', the log is enabled.
`/ncs-config/logs/ncs-log/file`	—
`/ncs-config/logs/ncs-log/ file/name (string)`	*name* is the full path to the actual log file.
`/ncs-config/logs/ncs-log/file/ enabled (boolean) [false]`	If 'true', file logging is enabled.
`/ncs-config/logs/ncs-log/syslog`	—
`/ncs-config/logs/ncs-log/ syslog/enabled (boolean) [false]`	If 'true', syslog messages are sent.
`/ncs-config/logs/ncs-log/ syslog/facility (daemon | authpriv | local0 | local1 | local2 | local3 | local4 | local5 | local6 | local7 | uint32)`	This optional value overrides the /ncs-config/logs/syslog-config/facility for the specified log.
`/ncs-config/logs/developer-log`	*developer-log* is a debug log for troubleshooting user-written Java code. Enable and check this log for problems with validation code. This log is enabled by default. In all other regards it can be configured as ncs-log. This log is not rotated; use logrotate(8).
`/ncs-config/logs/developer-log/ enabled (boolean) [true]`	If 'true', the log is enabled.
`/ncs-config/logs/developer-log/ file`	—
`/ncs-config/logs/developer-log/ file/name (string)`	*name* is the full path to the actual log file.
`/ncs-config/logs/developer-log/ file/enabled (boolean) [false]`	If 'true', file logging is enabled.
`/ncs-config/logs/developer-log/ syslog`	—
`/ncs-config/logs/developer-log/ syslog/enabled (boolean) [false]`	If 'true', syslog messages are sent.
`/ncs-config/logs/developer-log/ syslog/facility (daemon | authpriv | local0 | local1 | local2 | local3 | local4 | local5 | local6 | local7 | uint32)`	This optional value overrides the /ncs-config/logs/syslog-config/facility for the specified log.

Parameter	Description
`/ncs-config/logs/developer-log-level (error | info | trace) [info]`	Controls the level of developer messages to print in the developer log.
`/ncs-config/logs/audit-log`	*audit-log* is an audit log that records successful and failed logins to the WAE back plane. This log is enabled by default. In all other regards it can be configured as /ncs-config/logs/ncs-log. This log is not rotated; use logrotate(8).
`/ncs-config/logs/audit-log/ enabled (boolean) [true]`	If 'true', the log is enabled.
`/ncs-config/logs/audit-log/file`	—
`/ncs-config/logs/audit-log/ file/name (string)`	*name* is the full path to the actual log file.
`/ncs-config/logs/audit-log/ file/enabled (boolean) [false]`	If 'true', file logging is enabled.
`/ncs-config/logs/audit-log/ syslog`	—
`/ncs-config/logs/audit-log/ syslog/enabled (boolean) [false]`	If 'true', syslog messages are sent.
`/ncs-config/logs/audit-log/ syslog/facility (daemon | authpriv | local0 | local1 | local2 | local3 | local4 | local5 | local6 | local7 | uint32)`	This optional value overrides the /ncs-config/logs/syslog-config/facility for the specified log.
`/ncs-config/logs/audit-log-commit (boolean) [false]`	Controls whether the audit log should include messages about the resulting configuration changes for each commit to the running data store.
`/ncs-config/logs/netconf-log`	*netconf-log* is a log for troubleshooting northbound NETCONF operations, such as checking why a filter operation didn't return the data requested. This log is enabled by default. In all other regards it can be configured as /ncs-config/logs/ncs-log. This log is not rotated; use logrotate(8).
`/ncs-config/logs/netconf-log/ enabled (boolean) [true]`	If 'true', the log is enabled.
`/ncs-config/logs/netconf-log/ file`	—
`/ncs-config/logs/netconf-log/ file/name (string)`	*name* is the full path to the actual log file.
`/ncs-config/logs/netconf-log/ file/enabled (boolean) [false]`	If 'true', file logging is enabled.
`/ncs-config/logs/netconf-log/syslog`	—
`/ncs-config/logs/netconf-log/ syslog/enabled (boolean) [false]`	If 'true', syslog messages are sent.

Parameter	Description

`/ncs-config/logs/netconf-log/`
`syslog/facility (daemon | authpriv`
`| local0 | local1 | local2 | local3`
`| local4 | local5 | local6 | local7`
`| uint32)`
	This optional value overrides the /ncs-config/logs/syslog-config/facility for the specified log.

`/ncs-config/logs/snmp-log`	—
`/ncs-config/logs/snmp-log/ enabled` `(boolean) [true]`	If 'true', the log is enabled.
`/ncs-config/logs/snmp-log/file`	—
`/ncs-config/logs/snmp-log/` `file/name (string)`	*name* is the full path to the actual log file.
`/ncs-config/logs/snmp-log/` `file/enabled (boolean) [false]`	If 'true', file logging is enabled.
`/ncs-config/logs/snmp-log/ syslog`	—
`/ncs-config/logs/snmp-log/` `syslog/enabled (boolean) [false]`	If 'true', syslog messages are sent.

`/ncs-config/logs/snmp-log/`
`syslog/facility (daemon | authpriv`
`| local0 | local1 | local2 | local3`
`| local4 | local5 | local6 | local7`
`| uint32)`
	This optional value overrides the /ncs-config/logs/syslog-config/facility for the specified log.

`/ncs-config/logs/snmp-log-level` `(error | info) [info]`	Controls which level of SNMP PDUs are printed in the SNMP log. The value 'error' means that only PDUs with error-status not equal to 'noError' are printed.
`/ncs-config/logs/webui-browser-log`	*webui-browser-log* makes it possible to log Java script errors/exceptions in a log file on the target device instead of just in the browser's error console. This log is not enabled by default and is not rotated; use logrotate(8).
`/ncs-config/logs/webui-browser-log/` `enabled (boolean) [false]`	If 'true', the browser log is used.
`/ncs-config/logs/webui-browser-log/filename` `(string)`	This parameter is mandatory. The path to the filename where browser log entries are written.
`/ncs-config/logs/webui-access-log`	*webui-access-log* is an access log for the embedded WAE web server. This file adheres to the Common Log Format, as defined by Apache and others. This log is not enabled by default and is not rotated; use logrotate(8).
`/ncs-config/logs/webui-access-log/` `enabled (boolean) [false]`	If 'true', the access log is used.

Parameter	Description
`/ncs-config/logs/webui-access-log/` `traffic-log (boolean) [false]`	If 'true', all HTTP(S) traffic towards the embedded web server is logged in a log file named traffic.trace. This log is not enabled by default and is not rotated; use logrotate(8). **Caution** Do not use this log in a production setting.
`/ncs-config/logs/webui-access-log/` `dir (string)`	This parameter is mandatory. The path to the directory where the access log is written.
`/ncs-config/logs/netconf-trace-log`	*netconf-trace-log* is a log for understanding and troubleshooting northbound NETCONF protocol interactions. When this log is enabled, all NETCONF traffic to and from WAE is stored to a file. By default, all XML is pretty-printed. This slows down the NETCONF server, so be careful when enabling this log. This log is not rotated; use logrotate(8).
`/ncs-config/logs/netconf-trace-log/` `enabled (boolean) [false]`	If 'true', all NETCONF traffic is logged.
`/ncs-config/logs/netconf-trace-log/` `filename (string)`	This parameter is mandatory. The name of the file where the NETCONF traffic trace log is written.
`/ncs-config/logs/netconf-trace-log/` `format (pretty | raw) [pretty]`	The value 'pretty' means that the XML data is pretty-printed. The value 'raw' means that it is not pretty-printed.
`/ncs-config/logs/xpath-trace-log`	*xpath-trace-log* is a log for understanding and troubleshooting xpath evaluations. When this log is enabled, all xpath queries evaluated by WAE are logged to a file. This slows down WAE, so be careful when enabling this log. This log is not rotated; use logrotate(8).
`/ncs-config/logs/xpath-trace-log/` `enabled (boolean) [false]`	If 'true', all xpath execution is logged.
`/ncs-config/logs/xpath-trace-log/` `filename (string)`	This parameter is mandatory. The name of the file where the xpath trace log is written.
`/ncs-config/logs/error-log`	*error-log* is an error log used for internal logging from the WAE daemon. It is used for troubleshooting the WAE daemon itself, and should normally be disabled. This log is rotated by the WAE daemon.
`/ncs-config/logs/error-log/ enabled` `(boolean) [false]`	If 'true', error logging is performed.
`/ncs-config/logs/error-log/` `filename (string)`	This parameter is mandatory. *filename* is the full path to the actual log file. This parameter must be set if the error log is enabled.
`/ncs-config/logs/error-log/max-size` `(tailf:size) [S1M]`	*max-size* is the maximum size of an individual log file before it is rotated. Log filenames are reused when five logs have been exhausted.
`/ncs-config/logs/error-log/ debug`	—
`/ncs-config/logs/error-log/` `debug/enabled (boolean) [false]`	—

Parameter	Description
`/ncs-config/logs/error-log/` `debug/level (uint16) [2]`	—
`/ncs-config/logs/error-log/` `debug/tag (string)`	This parameter can be given multiple times.
`/ncs-config/candidate`	—
`/ncs-config/candidate/ filename` `(string)`	The candidate db-mode has been removed; this leaf no longer affects the WAE configuration. This leaf and the candidate container are retained for backward compatibility.
`/ncs-config/sort-transactions` `(boolean) [true]`	This parameter controls how WAE lists newly created, not yet committed list entries. If this value is set to 'false', WAE lists all new elements before listing existing data. If this value is set to 'true', WAE merges new and existing entries, and provides one sorted view of the data. This behavior works well when CDB is used to store configuration data, but if an external data provider is used, WAE does not know the sort order and cannot merge the new entries correctly. If an external data provider is used for configuration data, and if the sort order differs from CDB's sort order, this parameter should be set to 'false'.
`/ncs-config/enable-attributes` `(boolean) [true]`	This parameter controls whether WAE's attribute feature is enabled. There are two attributes: annotations and tags. These are available in northbound interfaces (the annotate command in the CLI, and the annotation XML attribute in NETCONF), but to be useful they need support from the underlying configuration data provider. CDB supports attributes, but if an external data provider is used for configuration data, and if it does not support the attribute callbacks, this parameter should be set to 'false'.
`/ncs-config/enable-inactive` `(boolean) [true]`	This parameter controls whether WAE's inactive feature is enabled. This feature also requires enableAttributes to be enabled. When WAE is used to control Juniper routers, this feature is required.
`/ncs-config/session-limits`	Limits concurrent access to WAE.
`/ncs-config/session-limits/max-sessions` `(uint32 | unbounded) [unbounded]`	Limits the total number of concurrent sessions to WAE.
`/ncs-config/session-limits/session-limit`	Limits concurrent access for a specific context to WAE. There can be multiple instances of this container element, each one specifying parameters for a specific context.
`/ncs-config/session-limits/` `session-limit/context (string)`	The context is cli, netconf, webui, snmp, or any other context string defined through the use of MAAPI. For example, if you use MAAPI to implement a CORBA interface to WAE, the MAAPI program could send the string 'corba' as context.

`/ncs-config/session-limits/`
`session-limit/max-sessions (uint32`
`| unbounded)`
	This parameter is mandatory. Limits the total number of concurrent sessions to WAE.

Parameter	Description
`/ncs-config/session-limits/ max-config-sessions (uint32 | unbounded) [unbounded]`	Limits the total number of concurrent configuration sessions to WAE.
`/ncs-config/session-limits/ config-session-limit`	Limits concurrent read-write transactions for a specific context to WAE. There can be multiple instances of this container element, each one specifying parameters for a specific context.
`/ncs-config/session-limits/ config-session-limit/context (string)`	The context is cli, netconf, webui, snmp, or any other context string defined through the use of MAAPI. For example, if you use MAAPI to implement a CORBA interface to WAE, the MAAPI program could send the string 'corba' as context.
`/ncs-config/session-limits/ config-session-limit/max-sessions (uint32 |unbounded)`	This parameter is mandatory. Limits the total number of concurrent configuration sessions to WAE for the corresponding context.
`/ncs-config/aaa`	—
`/ncs-config/aaa/ssh-login-grace-time (xs:duration) [PT10M]`	WAE servers close SSH connections after this time if the client has not successfully authenticated itself. If the value is 0, there is no time limit for client authentication. This is a global value for all SSH servers in WAE. Changing this value affects only SSH connections that are established after the change is made.
`/ncs-config/aaa/ssh-max-auth-tries (uint32 | unbounded) [unbounded]`	WAE servers close SSH connections when the client has made this number of unsuccessful authentication attempts. This is a global value for all SSH servers in WAE. Changing this value affects only SSH connections that are established after the change is made.
`/ncs-config/aaa/ssh-server-key-dir (string)`	*ssh-server-key-dir* is the directory file path where the keys used by the WAE SSH daemon are found. This parameter must be set if SSH is enabled for NETCONF or the CLI. If SSH is enabled, the server keys used by WAE are on the same format as the server keys used by openssh (that is, the same format as generated by 'ssh-keygen').
	Only DSA- and RSA-type keys can be used with the WAE SSH daemon, as generated by 'ssh-keygen' with the '-t dsa' and '-t rsa' switches, respectively. The key must be stored with an empty passphrase, and with the name 'ssh_host_dsa_key' if it is a DSA-type key, and with the name 'ssh_host_rsa_key' if it is an RSA-type key. The SSH server advertises support for those key types for which there is a key file available and for which the required algorithm is enabled. See the /ncs-config/ssh/algorithms/server-host-key leaf.

Parameter	Description
`/ncs-config/aaa/ssh-pubkey-authentication (none | local | system) [system]`	Controls how the WAE SSH daemon locates the user keys for public key authentication. If set to 'none', public key authentication is disabled. If set to 'local', and the user exists in /aaa/authentication/users, the keys in the user's 'ssh_keydir' directory are used. If set to 'system', the user is first looked up in /aaa/authentication/users, but only if /ncs-config/aaa/local-authentication/enabled is set to 'true'. If local-authentication is disabled, or if the user does not exist in /aaa/authentication/users but does exist in the OS password database, the keys in the user's $HOME/.ssh directory are used.
`/ncs-config/aaa/default-group (string)`	If the user group cannot be found in the AAA subsystem, a logged-in user ends up as a member of the default group (if specified). If a user logs in and the group membership cannot be established, the user has zero access rights.
`/ncs-config/aaa/auth-order (string)`	The default order for authentication is 'local-authentication pam external-authentication'. It is possible to change this order through this parameter.
`/ncs-config/aaa/expiration-warning (ignore | display | prompt) [ignore]`	When PAM or external authentication is used, the authentication mechanism might give a warning that the user's password is about to expire. This parameter controls how the WAE daemon processes that warning message. If set to 'ignore', the warning is ignored. If set to 'display', interactive user interfaces display the warning message at login. If set to 'prompt', interactive user interfaces display the warning message at login. The user must acknowledge the message before proceeding.
`/ncs-config/aaa/audit-user-name (always | known | never) [known]`	Controls the logging of the username when a failed authentication attempt is logged to the audit log. If set to "always", the username is always logged. If set to "known", the username is only logged when it is known to be valid (that is, when attempting local-authentication and the user exists in /aaa/authentication/users). Otherwise, it is logged as "[withheld]". If set to "never", the username is always logged as "[withheld]".
`/ncs-config/aaa/pam`	If PAM is used for login, the WAE daemon typically must run as root.
`/ncs-config/aaa/pam/enabled (boolean) [false]`	When set to 'true', WAE uses PAM for authentication.
`/ncs-config/aaa/pam/service (string) [common-auth]`	The PAM service to use for the login NETCONF/SSH CLI procedure. This can be any service installed in the /etc/pam.d directory. Different unices have different services installed under /etc/pam.d. Choose an existing service or create a new one.

Parameter	Description
`/ncs-config/aaa/pam/timeout (xs:duration) [PT10S]`	The maximum time that authentication waits for a reply from PAM. If the timeout is reached, the PAM authentication fails, but authentication attempts are made with other mechanisms as configured for /ncs-config/aaa/authOrder. The default is PT10S (10 seconds).
`/ncs-config/aaa/external-authentication`	—
`/ncs-config/aaa/external-authentication/enabled (boolean) [false]`	When set to 'true', external authentication is used.
`/ncs-config/aaa/external-authentication/executable (string)`	If external authentication is enabled, an executable on the local host can be launched to authenticate a user. The executable receives the username and the clear-text password on its standard input. The format is '[${USER};${PASS};]\n'. For example, if user is 'bob' and password is 'secret', the executable receives the line '[bob;secret;]' followed by a new line on its standard input. The program must parse this line. The task of the external program is to authenticate the user and also provide the user-to-groups mapping. If 'bob' is a member of the 'oper' and the 'lamers' groups, the program should echo 'accept oper lamers' on its standard output. If the user fails to authenticate, the program should echo 'reject ${reason}' on its standard output.
`/ncs-config/aaa/external-authentication/use-base64 (boolean) [false]`	When set to 'true', ${USER} and ${PASS} in the data passed to the executable are base64-encoded, allowing the password to contain ';' characters. For example, if user is 'bob' and password is 'secret', the executable receives the string '[Ym9i;c2VjcmV0;]' followed by a new line.
`/ncs-config/aaa/external-authentication/include-extra (boolean) [false]`	When set to 'true', additional information items are provided to the executable: source IP address and port, context, and protocol. The complete format is '[${USER};${PASS};${IP};$ {PORT};${CONTEXT};${PROTO};]\n'. Example: '[bob;secret;192.168.1.1;12345;cli;ssh;]\n'.
`/ncs-config/aaa/local-authentication`	—
`/ncs-config/aaa/local-authentication/enabled (boolean) [true]`	When set to 'true', WAE uses local authentication. The user data kept in the aaa namespace is used to authenticate users. When set to 'false', another authentication mechanism (such as PAM or external authentication) is used.
`/ncs-config/aaa/authentication-callback`	—
`/ncs-config/aaa/authentication-callback/ enabled (boolean) [false]`	When set to 'true', WAE invokes an application callback when authentication succeeds or fails. The callback might reject an otherwise successful authentication. If the callback has not been registered, all authentication attempts fail.
`/ncs-config/aaa/authorization`	—

Parameter	Description
`/ncs-config/aaa/authorization/enabled (boolean) [true]`	When set to 'false', all authorization checks are turned off, similar to the -noaaa flag in ncs_cli.
`/ncs-config/aaa/authorization/ callback`	—
`/ncs-config/aaa/authorization/callback/enabled (boolean) [false]`	When set to 'true', WAE invokes application callbacks for authorization. If the callbacks have not been registered, all authorization checks are rejected.
`/ncs-config/aaa/namespace (string) [http://tail-f.com/ns/aaa/1.1]`	To move the AAA data into another user-defined namespace, indicate that namespace here.
`/ncs-config/aaa/prefix (string) [/]`	To move the AAA data into another user-defined namespace, indicate the prefix path in that namespace where the WAE AAA namespace is mounted.
`/ncs-config/rollback`	Settings that control if and where rollback files are created. A rollback file contains a copy of the system configuration. The current running configuration is always stored in rollback0, the previous version in rollback1, and so on. The oldest saved configuration has the highest suffix.
`/ncs-config/rollback/ enabled (boolean) [false]`	When set to 'true', a rollback file is created whenever the running configuration is modified.
`/ncs-config/rollback/ directory (string)`	This parameter is mandatory. The location where rollback files are created.
`/ncs-config/rollback/ history-size (uint32) [35]`	The number of old configurations to save.
`/ncs-config/rollback/ type (delta) [delta]`	This parameter is deprecated. WAE supports only type 'delta'. It is not necessary to set a value for this parameter; it is retained only for backward compatibility. Type 'delta' means that only the changes are stored in the rollback file. Rollback file 0 contains the changes from the last configuration commit. This is space and time efficient for large configurations.
`/ncs-config/rollback/ rollback-numbering (rolling | fixed) [fixed]`	*rollback-numbering* is either 'fixed' or 'rolling'. If set to 'rolling', rollback file '0' always contains the last commit. If set to 'fixed', each rollback gets a unique increasing number.
`/ncs-config/ssh`	Controls the behavior of the SSH server built into WAE.
`/ncs-config/ssh/idle-connection-timeout (xs:duration) [PT10M]`	The maximum time that an authenticated connection to the SSH server is allowed to exist without open channels. If the timeout is reached, the SSH server closes the connection. The default is PT10M (10 minutes). A value of 0 means there is no timeout.
`/ncs-config/ssh/algorithms`	Defines custom lists of algorithms to be usable with the built-in SSH implementation. For each type of algorithm, an empty value means that all supported algorithms should be usable. A non-empty value (a comma-separated list of algorithm names) means that the intersection of the supported algorithms and the configured algorithms should be usable.

Parameter	Description
`/ncs-config/ssh/algorithms/server-host-key (string) []`	The supported serverHostKey algorithms (if implemented in libcrypto) are "ssh-dss" and "ssh-rsa", but for any SSH server, it is limited to those algorithms for which there is a host key installed in the directory given by /ncs-config/aaa/ssh-server-key-dir. To limit the usable serverHostKey algorithms to "ssh-dss", set this value to "ssh-dss" or avoid installing a key of any other type than ssh-dss in the sshServerKeyDir.
`/ncs-config/ssh/algorithms/kex (string) []`	The supported key exchange algorithms (as long as their hash functions are implemented in libcrypto) are "diffie-hellman-group-exchange-sha256", "diffie-hellman-group-exchange-sha1", "diffie-hellmangroup14-sha1", and "diffie-hellman-group1-sha1". To limit the usable key exchange algorithms to "diffie-hellman-group14-sha1" and "diffie-hellmangroup-exchange-sha256" (in that order), set this value to "diffie-hellman-group14-sha1, diffie-hellmangroup-exchange-sha256".
`/ncs-config/ssh/algorithms/dh-group`	The range of allowed group size the SSH server responds to the client during a "diffie-hellman-groupexchange". The range is the intersection of what the client requests. If there is none, the key exchange is terminated.
`/ncs-config/ssh/algorithms/ dh-group/min-size (dh-group-size-type) [2048]`	Minimum size of p, in bits.
`/ncs-config/ssh/algorithms/dh-group/max-size (dh-group-size-type) [4096]`	Maximum size of p, in bits.
`/ncs-config/ssh/algorithms/mac (string) []`	The supported mac algorithms (if implemented in libcrypto) are "hmac-md5", "hmac-sha1", "hmacsha2-256", "hmac-sha2-512", "hmac-sha1-96", and "hmac-md5-96".
`/ncs-config/ssh/algorithms/encryption (string) []`	The supported encryption algorithms (if implemented in libcrypto) are "aes128-ctr", "aes192-ctr", "aes256-ctr", "aes128-cbc", "aes256-cbc", and "3des-cbc".
`/ncs-config/ssh/client-alive-interval (xs:duration | infinity) [infinity]`	If no data has been received from a connected client for this long, a request that requires a response from the client is sent over the SSH transport.
`/ncs-config/ssh/client-alive-count-max (uint32) [3]`	If no data has been received from the client after this many consecutive client-alive-intervals have passed, the connection drops.
`/ncs-config/cli`	CLI parameters.
`/ncs-config/cli/enabled (boolean) [true]`	If 'true', the CLI server is started.
`/ncs-config/cli/allow-implicit-wildcard (boolean) [true]`	If 'true', users do not need to explicitly type * in the place of keys in lists, in order to see all list instances. If 'false', users must explicitly type * to see all list instances.
`/ncs-config/cli/completion-show-max (cli-max) [100]`	The maximum number of possible alternatives to present when doing completion.

Parameter	Description
`/ncs-config/cli/style (j | c)`	Style is either 'j' or 'c'. If set to 'j', the CLI is presented as a Juniper-style CLI. If 'c', the CLI appears as Cisco XR style.
`/ncs-config/cli/ssh`	—
`/ncs-config/cli/ssh/enabled (boolean) [true]`	*enabled* is either 'true' or 'false'. If 'true', the WAE CLI uses the built-in SSH server.
`/ncs-config/cli/ssh/ip (ipv4-address | ipv6-address) [0.0.0.0]`	*ip* is an IP address that the WAE CLI listens on for SSH connections. 0.0.0.0 means that it listens on the port (/ncs-config/cli/ssh/port) for all IPv4 addresses on the machine.
`/ncs-config/cli/ssh/port (port-number) [2024]`	The port number for CLI SSH.
`/ncs-config/cli/ssh/banner (string) []`	*banner* is a string that is presented to the client before authenticating when logging in to the CLI via the built-in SSH server.
`/ncs-config/cli/ssh/banner-file (string) []`	*banner-file* is the name of a file whose contents are presented (after any string given by the banner directive) to the client before authenticating when logging in to the CLI via the built-in SSH server.
`/ncs-config/cli/ssh/extra-listen`	A list of additional IP address and port pairs that the WAE CLI listens on for SSH connections.
`/ncs-config/cli/ssh/extra-listen/ip (ipv4-address | ipv6-address)`	—
`/ncs-config/cli/ssh/extra-listen/port (port-number)`	—
`/ncs-config/cli/top-level-cmds-in-sub-mode (boolean) [false]`	topLevelCmdsInSubMode is 'true' or 'false'. If 'true', all top-level commands in I and C style CLI are available in submodes.
`/ncs-config/cli/completion-meta-info (false | alt1 | alt2) [false]`	completionMetaInfo is 'false', 'alt1', or 'alt2'. If set to 'alt1', the alternatives shown for possible completions are prefixed as follows: containers with > lists with + leaf-lists + For example: Possible completions: ... > applications + apply-groups ... + dns-servers ... If set to 'alt2', possible completions are prefixed as follows: containers with > lists with children with +> lists without children + For example: Possible completions: ... > applications +>apply-groups ... + dns-servers ...
`/ncs-config/cli/allow-abbrev-keys (boolean) [false]`	allowAbbrevKeys is 'true' or 'false'. If 'false', key elements are not allowed to be abbreviated in the CLI. This is relevant in the J-style CLI when using the commands 'delete' and 'edit'. This is relevant in the C/I-style CLIs when using the commands 'no', 'show configuration', and for commands to enter submodes.

Parameter	Description
`/ncs-config/cli/j-align-leaf-values (boolean) [true]`	j-align-leaf-values is 'true' or 'false'. If 'true', the leaf values of all siblings in a container or list are aligned.
`/ncs-config/cli/enter-submode-on-leaf (boolean) [true]`	enterSubmodeOnLeaf is 'true' or 'false'. If 'true' (the default), setting a leaf in a submode from a parent mode results in entering the submode after the command has completed. If 'false', an explicit command for entering the submode is required—for example, if running the command **interface FastEthernet 1/1/1 mtu 1400** from the top level in config mode. If enterSubmodeOnLeaf is 'true', the CLI ends up in the 'interface FastEthernet 1/1/1' submode after the command execution. If 'false', the CLI remains at the top level. To enter the submode when set to 'false', the command **interface FastEthernet 1/1/1** is required. Applied to the C-style CLI.
`/ncs-config/cli/table-look-ahead (int64) [50]`	The tableLookAhead element tells confd how many rows to pre-fetch when displaying a table. The prefetched rows are used to calculate the required column widths for the table. If set to a small number, you should explicitly configure the column widths in the clispec file.
`/ncs-config/cli/more-buffer-lines (uint32 | unbounded) [unbounded]`	moreBufferLines is used to limit the buffering done by the more process. It can be 'unbounded' or a positive integer that describes the maximum number of lines to buffer.
`/ncs-config/cli/show-all-ns (boolean) [false]`	If showAllNs is 'true', all elem names are prefixed with the namespace prefix in the CLI. This is visible when setting values and when showing the configuration.
`/ncs-config/cli/suppress-fast-show (boolean) [false]`	suppressFastShow is 'true' or 'false'. If 'true', the fast show optimization is suppressed in the C-style CLI. The fast show optimization is somewhat experimental and might break certain operations.
`/ncs-config/cli/use-expose-ns-prefix (boolean) [true]`	If 'true', all nodes annotated with the tailf:cli-expose-ns-prefix result in the namespace prefix being shown/required. If 'false', the tailf:cli-expose-ns-prefix annotation is ignored. The container /devices/device/config has this annotation.
`/ncs-config/cli/show-defaults (boolean) [false]`	*show-defaults* is 'true' or 'false'. If 'true', default values are shown when displaying the configuration. The default value is shown inside a comment on the same line as the value. Showing default values can also be enabled in the CLI per session using the operational mode command **set show defaults true**.
`/ncs-config/cli/default-prefix (string) []`	*default-prefix* is a string that is placed in front of the default value when a configuration is shown with default values as comments.
`/ncs-config/cli/commit-retry-timeout (xs:duration | infinity) [PT0S]`	The commit timeout in the CLI. This timeout controls for how long the commit operation tries to complete the operation when some other entity is locking the database. A similar configuration parameter, /ncs-config/commit-retry-timeout, sets a timeout for WAE transactions in the JSON-RPC API.
`/ncs-config/cli/timezone (utc | local) [local]`	Time in the CLI can be local (as configured on the host) or UTC.

Parameter	Description
`/ncs-config/cli/with-defaults (boolean) [false]`	*with-defaults* is 'true' or 'false'. If 'false', leaf nodes that have their default values are not shown when the user displays the configuration, unless the user gives the 'details' option to the 'show' command. This is useful when there are many settings that are seldom used. If 'false', only the values actually modified by the user are shown.
`/ncs-config/cli/banner (string) []`	Banner shown to the user when the CLI is started. The default is empty.
`/ncs-config/cli/banner-file (string) []`	File whose contents are shown to the user (after any string set by the 'banner' directive) when the CLI is started. The default is empty.
`/ncs-config/cli/prompt1 (string) [\u@\h\M> ]`	Prompt used in operational mode. The string might contain a number of backslash-escaped special characters that are decoded as follows: • \d—Date in 'YYYY-MM-DD' format (for example, '2006-01-18'). • \h—Hostname up to the first '.' (or delimiter as defined by promptHostnameDelimiter). • \H—Current time in 24-hour HH:MM:SS format. • \T—Current time in 12-hour HH:MM:SS format. • \@—Current time in 12-hour am/pm format. • \A—Current time in 24-hour HH:MM format. • \u—Username of the current user. • \m—Mode name (only used in XR style). • \M—Mode name inside parenthesis if in a mode.
`/ncs-config/cli/prompt2 (string) [\u@\h\M% ]`	Prompt used in configuration mode. The string might contain a number of backslash-escaped special characters that are decoded as described for prompt1.
`/ncs-config/cli/c-prompt1 (string) [\u@\h\M> ]`	Prompt used in operational mode in the Cisco XR-style CLI. The string might contain a number of backslash-escaped special characters that are decoded as described for prompt1.
`/ncs-config/cli/c-prompt2 (string) [\u@\h\M% ]`	Prompt used in configuration mode in the Cisco XR-style CLI. The string might contain a number of backslash-escaped special characters that are decoded as described for prompt1.
`/ncs-config/cli/prompt-hostname-delimiter (string) [.]`	When the \h token is used in a prompt, the first part of the hostname up until the first occurrence of the promptHostnameDelimiter is used.
`/ncs-config/cli/show-log-directory (string) [/var/log]`	Location where the **show log** command looks for log files.
`/ncs-config/cli/idle-timeout (xs:duration) [PT30M]`	Maximum idle time before terminating a CLI session. The default is PT30M (30 minutes).

Parameter	Description
`/ncs-config/cli/prompt-sessions-cli` `(boolean) [false]`	promptSessionsCLI is 'true' or 'false'. If 'true', only the current CLI sessions are displayed when the user tries to start a new CLI session and the maximum number of sessions has been reached. Note that MAAPI sessions with their context set to 'cli' are regarded as CLI sessions and are listed as such.
`/ncs-config/cli/suppress-` `ned-errors (boolean) [false]`	Suppress errors from NED devices. Make log-communication between WAE and its devices more silent. Be careful with this option, because it might suppress interesting errors as well.
`/ncs-config/cli/disable-idle-timeout-on-cmd` `(boolean) [true]`	*disable-idle-timeout-on-cmd* is 'true' or 'false'. If 'false', the idle timeout triggers even when a command is running in the CLI. If 'true', the idle timeout only triggers if the user is idling at the CLI prompt.
`/ncs-config/cli/command-timeout` `(xs:duration | infinity) [infinity]`	Global command timeout: terminate the command unless the command has completed within the timeout. We do not recommend using this feature because it might have undesirable effects in a loaded system where normal commands take longer to complete. This timeout can be overridden by a command-specific timeout specified in the ncs.cli file.
`/ncs-config/cli/space-completion`	—
`/ncs-config/cli/space-completion/enabled` `(boolean)`	—
`/ncs-config/cli/ignore-leading-whitespace` `(boolean)`	If 'false', the CLI shows completion help when you enter TAB or SPACE as the first characters on a row. If 'true', leading SPACE and TAB are ignored. Enter '?' for a list of possible alternatives. Setting the value to 'true' makes it easier to paste scripts into the CLI.
`/ncs-config/cli/auto-wizard`	The default value for autowizard in the CLI. Users can always enable or disable the autowizard in each session; this controls the initial session value.
`/ncs-config/cli/auto-wizard/enabled` `(boolean) [true]`	**enabled** is 'true' or 'false'. If 'true', the CLI prompts the user for required attributes when a new identifier is created.
`/ncs-config/cli/restricted-file-access` `(boolean) [false]`	*restricted-file-access* is 'true' or 'false'. If 'true', a CLI user cannot access files and directories outside the home directory tree.
`/ncs-config/cli/restricted-file-regexp` `(string) []`	*restricted-file-regexp* is either an empty string or a regular expression (AWK style). If not empty, all files and directories created or accessed must match the regular expression. This can be used to ensure that certain symbols do not occur in created files.
`/ncs-config/cli/history-save` `(boolean) [true]`	If 'true', the CLI history is saved between CLI sessions. The history is stored in the state directory.
`/ncs-config/cli/history-remove-duplicates` `(boolean) [false]`	If 'true', repeated commands in the CLI are only stored once in the history. Each invocation of the command only updates the date of the last entry. If 'false', duplicates are stored in the history.
`/ncs-config/cli/history-max-size` `(int64) [1000]`	Sets the maximum configurable history size.

Parameter	Description
`/ncs-config/cli/message-max-size (int64) [10000]`	Sets the maximum size of user messages.
`/ncs-config/cli/show-commit-progress (boolean) [true]`	*show-commit-progress* is 'true' or 'false'. If 'true', the commit operation in the CLI provides progress information.
`/ncs-config/cli/commit-message (boolean) [true]`	CLI prints a message when a commit is executed.
`/ncs-config/cli/use-double-dot-ranges (boolean) [true]`	*use-double-dot-ranges* is 'true' or 'false'. If 'true', range expressions are given as 1..3. If 'false', ranges are given as 1-3.
`/ncs-config/cli/allow-range-expression-all-types (boolean) [true]`	*allow-range-expression-all-types* is 'true' or 'false'. If 'true', range expressions are allowed for all key values regardless of type.
`/ncs-config/cli/suppress-range-keyword (boolean) [false]`	*suppress-range-keyword* is 'true' or 'false'. If 'true', the 'range' keyword is not allowed in C- and I-style for range expressions.
`/ncs-config/cli/commit-message-format (string) [ System message at $(time)... Commit performed by $(user) via $(proto) using $(ctx). ]`	The format of the CLI commit messages.
`/ncs-config/cli/suppress-commit-message-context (string)`	This parameter can be given multiple times. A list of contexts for which a commit message is not displayed. A good value is [system], which makes all system-generated commits go unnoticed in the CLI. A context is either the name of an agent (CLI, web UI, NETCONF, SNMP) or a free-form text string if the transaction is initiated from MAAPI.
`/ncs-config/cli/show-subsystem-messages (boolean) [true]`	*show-subsystem-messages* is 'true' or 'false'. If 'true', the CLI displays a system message whenever a connected daemon starts or stops.
`/ncs-config/cli/show-editors (boolean) [true]`	*show-editors* is 'true' or 'false'. If 'true', a list of current editors is displayed when a user enters configure mode.
`/ncs-config/cli/rollback-aaa (boolean) [false]`	If 'true', AAA rules are applied when a rollback file is loaded. Rollback might not be possible if other users made changes that the current user does not have access privileges to.
`/ncs-config/cli/rollback-numbering (rolling | fixed) [fixed]`	*rollback-numbering* is 'fixed' or 'rolling'. If 'rolling', rollback file '0' always contains the last commit. If 'fixed', each rollback gets a unique increasing number.
`/ncs-config/cli/show-service-meta-data (boolean) [false]`	If 'true', backpointers and refcounts are displayed by default when showing the configuration. The default can be overridden by the pipe flags 'display service-meta' and 'hide service-meta'.
`/ncs-config/rest`	Controls how the embedded WAE web server should behave with respect to TCP and SSL.

Parameter	Description
`/ncs-config/rest/enabled (boolean) [false]`	*enabled* is 'true' or 'false'. If 'true', the web server is started.
`/ncs-config/rest/custom-headers`	—
`/ncs-config/rest/custom-headers/header`	—
`/ncs-config/rest/custom-headers/header/name (string)`	—
`/ncs-config/rest/custom-headers/header/value (string)`	This parameter is mandatory.
`/ncs-config/restconf`	Controls settings for the RESTCONF API.
`/ncs-config/restconf/enabled (boolean) [false]`	*enabled* is 'true' or 'false'. If 'true', the RESTCONF API is enabled on the web server used by the web UI. Note that the web UI must also be enabled.
`/ncs-config/restconf/root-resource (string) [restconf]`	The RESTCONF root resource path.
`/ncs-config/webui`	Controls how the embedded WAE web server should behave with respect to TCP and SSL.
`/ncs-config/webui/custom-headers`	*custom-headers* contains any number of header elements, with a valid header-field as defined in RFC7230. The headers are part of HTTP responses on '/login.html', '/index.html', and '/jsonrpc'.
`/ncs-config/webui/custom-headers/header`	—
`/ncs-config/webui/custom-headers/header/name (string)`	—
`/ncs-config/webui/custom-headers/header/value (string)`	This parameter is mandatory.
`/ncs-config/webui/enabled (boolean) [false]`	*enabled* is 'true' or 'false'. If 'true', the web server is started.
`/ncs-config/webui/server-name (string) [localhost]`	The hostname that the web server serves.
`/ncs-config/webui/match-host-name (boolean) [false]`	Specifies whether the web server should only serve URLs that adhere to the server-name defined above. By default, the server-name is 'localhost' and match-host-name is 'false'; any server name can be given in the URL. If you want the server to only accept URLs that adhere to the server-name, enable this setting.
`/ncs-config/webui/cache-refresh-secs (uint64) [0]`	The WAE web server uses a RAM cache for static content. An entry sits in the cache for a number of seconds before it is reread from disk (on access). The default is 0.

Parameter	Description
`/ncs-config/webui/max-ref-entries (uint64) [100]`	Leafref and keyref entries are represented as drop-down menus in the automatically generated web UI. By default, no more than 100 entries are fetched. This element makes this number configurable.
`/ncs-config/webui/docroot (string)`	The location of the document root on disk. If this configurable is omitted, the docroot points instead to the next generation docroot in the WAE distribution.
`/ncs-config/webui/login-dir (string)`	*login-dir* points out an alternative login directory that contains the HTML code used to log in to the web UI. This directory is mapped to https://<*ip-address*>/login. If this element is not specified, the default login/ directory in the docroot is used instead.
`/ncs-config/webui/X-Frame-Options (DENY | SAMEORIGIN | ALLOW-FROM) [DENY]`	By default the *X-Frame-Options* header is set to DENY for the /login.html and /index.html pages. With this header, you can set it to SAMEORIGIN or ALLOW-FROM instead.
`/ncs-config/webui/disable-auth`	—
`/ncs-config/webui/disable-auth/dir (string)`	This parameter can be given multiple times. The *disable-auth* element contains any number of *dir* elements. Each *dir* element points to a directory path in the docroot that should not be restricted by the AAA engine. If no *dir* elements are specified, the following directories and files are not restricted by the AAA engine: '/login' and '/login.html'.
`/ncs-config/webui/allow-symlinks (boolean) [true]`	Allows symlinks in the docroot directory.
`/ncs-config/webui/transport`	Controls which transport services (for example, TCP or SSL) the web server should listen on.
`/ncs-config/webui/transport/tcp`	Controls how the web server TCP transport service should behave.
`/ncs-config/webui/transport/tcp/enabled (boolean) [true]`	*enabled* is 'true' or 'false'. If 'true', the web server uses clear text TCP as a transport service.
`/ncs-config/webui/transport/tcp/ redirect (string)`	Redirects the user to the specified URL. Two macros can be specified: @HOST@ and @PORT@. For example: https://@HOST@:443 or https://192.12.4.3:@PORT@
`/ncs-config/webui/transport/ tcp/ip (ipv4-address | ipv6-address) [0.0.0.0]`	The IP address that the web server should listen on. 0.0.0.0 means that it listens on the port (/ncsconfig/webui/transport/tcp/port) for all IPv4 addresses on the machine.
`/ncs-config/webui/transport/ tcp/port (port-number) [8008]`	*port* is a valid port number to use in combination with the address in /ncs-config/webui/transport/tcp/ip.
`/ncs-config/webui/transport/tcp/extra-listen`	A list of additional IP address and port pairs that the web server should also listen on.

Parameter	Description
`/ncs-config/webui/ transport/tcp/extra-listen/ip (ipv4-address | ipv6-address)`	—
`/ncs-config/webui/ transport/tcp/extra-listen/port (port-number)`	—
`/ncs-config/webui/ transport/ssl`	Controls how the web server SSL transport service should behave. SSL is widely deployed on the Internet; virtually all online shopping and bank transactions are done with SSL encryption. There are many good sources that describe SSL in detail; for example, http://www.tldp.org/HOWTO/SSL-Certificates-HOWTO/ describes how to manage certificates and keys.
`/ncs-config/webui/ transport/ssl/enabled (boolean) [false]`	*enabled* is 'true' or 'false'. If 'true', the web server uses SSL as a transport service.
`/ncs-config/webui/transport/ ssl/redirect (string)`	Redirects the user to the specified URL. Two macros can be specified: @HOST@ and @PORT@. For example: http://@HOST@:80 or http://192.12.4.3:@PORT@
`/ncs-config/webui/transport/ssl/ip (ipv4-address | ipv6-address) [0.0.0.0]`	The IP address on which the web server listens for incoming SSL connections. 0.0.0.0 means that it listens on the port (/ncs-config/webui/transport/ssl/port) for all IPv4 addresses on the machine.
`/ncs-config/webui/ transport/ssl/port (port-number) [8888]`	*port* is a valid port number to use in combination with /ncs-config/webui/transport/ssl/ip.
`/ncs-config/webui/transport/ssl/extra-listen`	A list of additional IP address and port pairs on which the web server listens for incoming SSL connections.
`/ncs-config/webui/ transport/ssl/extra-listen/ip (ipv4-address | ipv6-address)`	—
`/ncs-config/webui/ transport/ssl/extra-listen/port (port-number)`	—
`/ncs-config/webui/transport/ ssl/key-file (string)`	Specifies the file that contains the private key for the certificate. Read more about certificates in /ncs-config/webui/ transport/ssl/cert-file. If this configurable is omitted, the keyFile points instead to a built-in, self-signed certificate/key in the WAE distribution. Note: Only use this certificate/key for test purposes.

Parameter	Description
`/ncs-config/webui/transport/ ssl/cert-file (string)`	Specifies the file that contains the server certificate. The certificate is either a self-signed test certificate or a genuine, validated certificate bought from a certificate authority (CA). If this configurable is omitted, the keyFile points instead to a built-in, self-signed certificate/key in the WAE distribution. Note: Only use this certificate/key for test purposes.
	The WAE distribution comes with a server certificate that can be used for testing (${NCS_DIR}/var/ncs/webui/ cert/host.{cert,key}). This server certificate has been generated using a local CA certificate:
	$ openssl OpenSSL> genrsa -out ca.key 4096 OpenSSL> req -new -x509 -days 3650 -key ca.key - out ca.cert OpenSSL> genrsa -out host.key 4096 OpenSSL> req -new -key host.key -out host.csr OpenSSL> x509 -req -days 365 -in host.csr -CA ca.cert \ -CAkey ca.key -set_serial 01 -out host.cert
`/ncs-config/webui/transport/ ssl/ca-cert-file (string)`	Specifies the file that contains the trusted certificates to use during client authentication and to use when attempting to build the server certificate chain. The list is also used in the list of acceptable CA certificates passed to the client when a certificate is requested.
	The WAE distribution comes with a CA certificate that can be used for testing (${NCS_DIR}/var/ncs/ webui/ca_cert/ca.cert). This CA certificate has been generated as shown above.
`/ncs-config/webui/transport/ ssl/verify (1 | 2 | 3) [1]`	Specifies the level of verification the server does on client certificates:
	• 1—No verification.
	• 2—The server asks the client for a certificate but does not fail if the client does not supply one.
	• 3—The server requires the client to supply a client certificate.
	If ca-cert-file has been set to the ca.cert file generated above, you can verify that it works by using:
	$ openssl s_client -connect 127.0.0.1:8888 \ -cert client.cert -key client.key
	For this to work, client.cert must have been generated using the ca.cert from above:
	$ openssl OpenSSL> genrsa -out client.key 4096 OpenSSL> req -new -key client.key -out client.csr OpenSSL> x509 -req -days 3650 -in client.csr -CA ca.cert \ -CAkey ca.key -set_serial 01 -out client.cert
`/ncs-config/webui/transport/ ssl/depth (uint64) [1]`	Specifies the depth of certificate chains the server is prepared to follow when verifying client certificates.

Parameter	Description
`/ncs-config/webui/transport/` `ssl/ciphers (string) [DEFAULT]`	Specifies the cipher suites for the server to use. The ciphers are a colon-separated list from the following set: ECDHEECDSA-AES256-SHA384, ECDHE-RSA-AES256-SHA384, ECDH-ECDSA-AES256-SHA384, ECDH-RSA-AES256-SHA384, DHE-RSA-AES256-SHA256, DHE-DSS-AES256-SHA256, AES256-SHA256, ECDHE-ECDSA-AES128-SHA256, ECDHE-RSA-AES128-SHA256, ECDHECDSA-AES128-SHA256, ECDH-RSA-AES128-SHA256, DHE-RSA-AES128-SHA256, DHEDSS-AES128-SHA256, AES128-SHA256, ECDHE-ECDSA-AES256-SHA, ECDHE-RSA-AES256-SHA, DHE-RSA-AES256-SHA, DHE-DSS-AES256-SHA, ECDH-ECDSA-AES256-SHA, ECDHRSA-AES256-SHA, AES256-SHA, ECDHE-ECDSA-DES-CBC3-SHA, ECDHE-RSA-DES-CBC3-SHA, EDH-RSA-DES-CBC3-SHA, EDH-DSS-DES-CBC3-SHA, ECDH-ECDSA-DES-CBC3-SHA, ECDH-RSA-DES-CBC3-SHA, DES-CBC3-SHA, ECDHE-ECDSA-AES128-SHA, ECDHE-RSAAES128-SHA, DHE-RSA-AES128-SHA, DHE-DSS-AES128-SHA, ECDH-ECDSA-AES128-SHA, ECDH-RSA-AES128-SHA, AES128-SHA, ECDHE-ECDSA-RC4-SHA, ECDHE-RSA-RC4-SHA, RC4-SHA, RC4-MD5, EDH-RSA-DES-CBC-SHA, ECDH-ECDSA-RC4-SHA, ECDH-RSA-RC4-SHA, and DES-CBC-SHA, or the word "DEFAULT" (use the listed set except the suites using DES, RC4, or MD5 algorithms) See the OpenSSL manual page ciphers(1) for the definition of the cipher suites. Note: The general cipher list syntax described in ciphers(1) is not supported.
`/ncs-config/webui/transport/` `ssl/protocols (string) [DEFAULT]`	Specifies the SSL/TLS protocol versions for the server to use as a whitespace-separated list from the set sslv3 tlsv1 tlsv1.1 tlsv1.2, or the word "DEFAULT" (use all supported protocol versions except sslv3).
`/ncs-config/webui/cgi`	CGI-script support.
`/ncs-config/webui/cgi/ enabled` `(boolean) [false]`	*enabled* is 'true' or 'false'. If 'true', CGI-script support is enabled.
`/ncs-config/webui/cgi/ dir (string)` `[cgi-bin]`	The directory path to the location of the CGI-scripts.
`/ncs-config/webui/cgi/` `request-filter (string)`	Specifies that characters not specified in the regexp should be filtered out silently.
`/ncs-config/webui/cgi/` `max-request-length (uint16)`	Specifies the maximum number of characters in a request. All characters that exceed this limit are silently ignored.
`/ncs-config/webui/cgi/php`	PHP support.
`/ncs-config/webui/cgi/php/ enabled` `(boolean) [false]`	*enabled* is 'true' or 'false'. If 'true', PHP support is enabled.
`/ncs-config/webui/ idle-timeout` `(xs:duration) [PT30M]`	The maximum idle time before terminating a web UI session. PT0M means no timeout. The default is PT30M (30 minutes).

Parameter	Description
`/ncs-config/webui/ absolute-timeout (xs:duration) [PT60M]`	The maximum absolute time before terminating a web UI session. PT0M means no timeout. The default is PT60M (60 minutes).
`/ncs-config/webui/ rate-limiting (uint64) [1000000]`	The maximum number of JSON-RPC requests allowed every hour. 0 means infinity. The default is 1 million.
`/ncs-config/webui/ audit (boolean) [true]`	*audit* is 'true' or 'false'. If 'true', JSON-RPC/CGI requests are logged to the audit log.
`/ncs-config/japi`	Java-API parameters.
`/ncs-config/japi/new-session-timeout (xs:duration) [PT30S]`	The timeout for a data provider to respond to a control socket request; see DpTrans. If the Dp fails to respond within the given time, it is disconnected.
`/ncs-config/japi/query-timeout (xs:duration) [PT120S]`	The timeout for a data provider to respond to a worker socket query; see DpTrans. If the Dp fails to respond within the given time, it is disconnected.
`/ncs-config/japi/connect-timeout (xs:duration) [PT60S]`	The timeout for a data provider to send an initial message after connecting the socket to the WAE server. If the Dp fails to initiate the connection within the given time, it is disconnected.
`/ncs-config/japi/object-cache-timeout (xs:duration) [PT2S]`	The timeout for the cache used by the getObject() and iterator(),nextObject() callback requests. WAE caches the result of these calls and serves getElem() requests from northbound agents from the cache. Setting this timeout too low causes the callbacks to be non-functional. For example, getObject() can be invoked for each getElem() request from a northbound agent.
`/ncs-config/japi/event-reply-timeout (xs:duration) [PT120S]`	The timeout for the reply from an event notification subscriber for a notification that requires a reply; see the Notif class. If the subscriber fails to reply within the given time, the event notification socket is closed.
`/ncs-config/netconf-north-bound`	Controls how the NETCONF agent should behave with respect to NETCONF and SSH.
`/ncs-config/netconf-north-bound/ enabled (boolean) [true]`	*enabled* is 'true' or 'false'. If 'true', the NETCONF agent is started.
`/ncs-config/netconf-north-bound/ transport`	Controls which transport services (TCP or SSH) the NETCONF agent should listen on.
`/ncs-config/netconf-north-bound/ transport/ssh`	Controls how the NETCONF SSH transport service should behave.
`/ncs-config/netconf-north-bound/ transport/ssh/enabled (boolean) [true]`	*enabled* is 'true' or 'false'. If 'true', the NETCONF agent uses SSH as a transport service.
`/ncs-config/netconf-north-bound/ transport/ssh/ip (ipv4-address | ipv6-address) [0.0.0.0]`	*ip* is an IP address that the WAE NETCONF agent listens on. 0.0.0.0 means that it listens on the port (/ncs-config/netconf-north-bound/transport/ssh/port) for all IPv4 addresses on the machine.

Parameter	Description
`/ncs-config/netconf-north-bound/transport/ssh/port (port-number) [2022]`	*port* is a valid port number to use in combination with /ncs-config/netconf-north-bound/transport/ssh/ip. The standard port for NETCONF over SSH is 830.
`/ncs-config/netconf-north-bound/transport/ssh/extra-listen`	A list of additional IP address and port pairs that the WAE NETCONF agent listens on.
`/ncs-config/netconf-north-bound/transport/ssh/extra-listen/ip (ipv4-address | ipv6-address)`	—
`/ncs-config/netconf-north-bound/transport/ssh/extra-listen/port (port-number)`	—
`/ncs-config/netconf-north-bound/transport/tcp`	NETCONF over TCP is not standardized, but it can be useful during development (for example, to use netcat for scripting). It is also useful when using your own proprietary transport. You can set up the NETCONF agent to listen on localhost and then proxy it from your transport service module.
`/ncs-config/netconf-north-bound/transport/tcp/enabled (boolean) [false]`	*enabled* is 'true' or 'false'. If 'true', the NETCONF agent uses clear text TCP as a transport service.
`/ncs-config/netconf-north-bound/transport/tcp/ip (ipv4-address | ipv6-address) [0.0.0.0]`	*ip* is an IP address that the WAE NETCONF agent listens on. 0.0.0.0 means that it listens on the port (/ncs-config/netconf-north-bound/transport/tcp/port) for all IPv4 addresses on the machine.
`/ncs-config/netconf-north-bound/transport/tcp/port (port-number) [2023]`	*port* is a valid port number to use in combination with /ncs-config/netconf-north-bound/transport/tcp/ip.
`/ncs-config/netconf-north-bound/transport/tcp/extra-listen`	A list of additional IP address and port pairs that the WAE NETCONF agent listens on.
`/ncs-config/netconf-north-bound/transport/tcp/extra-listen/ip (ipv4-address | ipv6-address)`	—
`/ncs-config/netconf-north-bound/transport/tcp/extra-listen/port (portnumber)`	—
`/ncs-config/netconf-north-bound/extended-sessions (boolean) [false]`	If extended-sessions are enabled, all WAE sessions can be terminated using <kill-session>. Not only can other NETCONF sessions be terminated, but also CLI sessions, web UI sessions, and so on. If a session holds a lock, its session ID is returned in the <lock-denied>, instead of '0'. This extension is not covered by the NETCONF specification; therefore, it is false by default.

Parameter	Description
`/ncs-config/netconf-north-bound/idle-timeout (xs:duration) [PT0S]`	The maximum idle time before terminating a NETCONF session. If the session is waiting for notification or has a pending confirmed commit, the idle timeout is not used. The default value is 0, which means no timeout.
`/ncs-config/netconf-north-bound/rpc-errors (close | inline) [close]`	If *rpc-errors* is 'inline' and an error occurs during the processing of a <get> or <get-config> request when WAE tries to fetch data from a data provider, WAE generates an rpc-error element in the faulty element, and continue to process the next element. If an error occurs and *rpc-errors* is 'close', WAE closes the NETCONF transport.
`/ncs-config/netconf-north-bound/max-batch-processes (uint32 | unbounded) [unbounded]`	Controls the number of concurrent NETCONF batch processes. A batch process can be started by the agent if a new NETCONF operation is implemented as a batch operation.
`/ncs-config/netconf-north-bound/capabilities`	Controls which NETCONF capabilities to enable.
`/ncs-config/netconf-north-bound/capabilities/url`	Turns on the URL capability options to support.
`/ncs-config/netconf-north-bound/capabilities/url/enabled (boolean) [false]`	*enabled* is 'true' or 'false'. If 'true', the URL NETCONF capability is enabled.
`/ncs-config/netconf-north-bound/capabilities/url/file`	Controls how the URL file support should behave.
`/ncs-config/netconf-north-bound/capabilities/url/file/enabled (boolean) [true]`	*enabled* is 'true' or 'false'. If 'true', the URL file scheme is enabled.
`/ncs-config/netconf-north-bound/capabilities/url/file/root-dir (string)`	*root-dir* is a directory path on disk where ConfD stores the result from an NETCONF operation using the URL capability. This parameter must be set if the file URL scheme is enabled.
`/ncs-config/netconf-north-bound/capabilities/url/ftp`	Controls how the URL FTP scheme should behave.
`/ncs-config/netconf-north-bound/capabilities/url/ftp/enabled (boolean) [true]`	*enabled* is 'true' or 'false'. If 'true', the URL FTP scheme is enabled.
`/ncs-config/netconf-north-bound/capabilities/url/sftp`	Controls how the URL SFTP scheme should behave.
`/ncs-config/netconf-north-bound/capabilities/url/sftp/enabled (boolean) [true]`	*enabled* is 'true' or 'false'. If 'true', the URL SFTP scheme is enabled.
`/ncs-config/netconf-north-bound/capabilities/inactive`	Controls the inactive capability option.

Parameter	Description
`/ncs-config/netconf-north-bound/ capabilities/inactive/enabled (boolean) [true]`	*enabled* is 'true' or 'false'. If 'true', the 'http://tail-f.com/ns/netconf/inactive/1.0' capability is enabled.
`/ncs-config/southbound-source-address`	Specifies the source address to use for southbound connections from WAE to devices. In most cases the source address assignment is best left to the TCP/IP stack in the OS, because an incorrect address might result in connection failures. However, if the stack could choose more than one address, and you need to restrict the choice to one address, these settings can be used.
`/ncs-config/southbound-source-address/ ipv4 (ipv4-address)`	The source address to use for southbound IPv4 connections. If not set, the source address is assigned by the OS.
`/ncs-config/southbound-source-address/ ipv6 (ipv6-address)`	The source address to use for southbound IPv6 connections. If not set, the source address is assigned by the OS.
`/ncs-config/ha`	—
`/ncs-config/ha/enabled (boolean) [false]`	If 'true', HA mode is enabled.
`/ncs-config/ha/ip (ipv4-address | ipv6-address) [0.0.0.0]`	The IP address that WAE listens to for incoming connections from other HA nodes.
`/ncs-config/ha/port (port-number) [4570]`	The port number that WAE listens to for incoming connections from other HA nodes.
`/ncs-config/ha/tick-timeout (xs:duration) [PT20S]`	Defines the timeout between keepalive ticks sent between HA nodes. The value 'PT0' means that no keepalive ticks are ever sent.
`/ncs-config/scripts`	It is possible to add scripts to control various things in WAE, such as post-commit callbacks. New CLI commands can also be added. The scripts must be stored under /ncs-config/scripts/dir, where there is a subdirectory for each script category. For some script categories it suffices to add a script in the correct subdirectory to enable the script. For others some configuration must be done.
`/ncs-config/scripts/dir (string)`	This parameter can be given multiple times. The directory path to the location of plug-and-play scripts. The scripts directory must have the following subdirectories: scripts/command/ post-commit/
`/ncs-config/large-scale`	—
`/ncs-config/large-scale/lsa`	—
`/ncs-config/large-scale/lsa/enabled (boolean) [false]`	Enables Layered Service Architecture (LSA), which requires a separate Cisco Smart License.