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 8.

Collection or Capability	NIMO	Description	Prerequisite/Notes
Network Collection NIMOs			
Topology Collection Using the IGP Database, on page 4	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 5	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. • 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 12	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 16	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 15	• port-cfg-parse-nimo • srlg-cfg-parse-nimo • vpn-cfg-parse-nimo • lsp-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.
Multilayer (L3-L1) Collection	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 .
LSP Configuration Collection, on page 13	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 18	lsp-snmp-nimo	Discovers LSPs using SNMP.	A network model with basic topology collection must exist.
LSP Collection Using XTC, on page 14	lsp-pcep-xtc-nimo	Discovers PCEP LSPs using XTC.	The Topology Collection Using XTC, on page 5 must be completed before running this collection.

Collection or Capability	NIMO	Description	Prerequisite/Notes
Segment Routing Traffic Matrix Collection, on page 11	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 8	—	Aggregates various NIMO information into a single consolidated network model.	Configured network models with information you want to merge into one final network model.
Autonomous System (AS) Model Consolidation, on page 10	as-merger	Resolves interfaces, circuits, etc., that are shared between AS models to create a single, consolidated network model.	• 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 26	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 18. • If collecting VPN traffic, a network model with VPN collection must exist. See VPN Collection, on page 12.
Network Model Layout (Visualization), on page 29	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.
Demand Mesh Creation, on page 31	demandmesh-creator-nimo	Creates a demand mesh between a set of source and destination nodes.	A network model with basic topology collection must exist.
Demand Deduction, on page 32	demand-deduction-nimo	Runs demand deduction using measured SNMP traffic and by applying demand mesh builds of end-to-end demands (traffic matrix) to the network model.	This NIMO requires that you use a consolidated network model (aggregator) as a source network that includes demands and interface measurements (traffic-poller).

Collection or Capability	NIMO	Description	Prerequisite/Notes
Running External Scripts Against a Network Model, on page 33	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 8.

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 and, although not common, topology discovery without interfaces or port detail collection. 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.
- 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.

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 enter the applicable IP addresses of the nodes.

a) Select one of the following from the node-filter list:

- INCLUDE ONLY—Include only the nodes specified in the node-filter-list tab.
- EXCLUDE ONLY—Exclude only the nodes specified in the node-filter-list tab.
- IGNORE FILTER—Include all nodes during collection.

b) To enter nodes, click the **node-filter-list** link.

c) Click the plus (+) sign to enter a node, then click **Add**.

d) Navigate back to the node-filter-list tab (wae:networks/network/<*network_model_name*>/nimo/topo-igp-nimo:topo-igp-nimo/igp-config/<*IGP_index*>/node-filter) to enter multiple nodes.

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 1.

Topology Collection Using XTC

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.

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 and Configure Network Access.
- An XTC agent must be configured and running. For more information, see Configuring XTC Agents Using the Expert Mode.

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 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 7. From the WAE UI, you can also hover your mouse over each field to view tooltips.

Step 7 (Optional) To exclude individual nodes from collection, click the **node-blacklist** tab and enter the applicable IP addresses of the nodes. For example, you might not want to see the XTC nodes in the 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 1.

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.
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.

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.

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

Step 1 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.

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.

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

Step 8 (Optional) Rules are automatically generated when a source is added or removed from the source list, or when a refresh is invoked in the resulting aggregator (dependent on the capabilities reported by source NIMOs). To select a rule set other than the default, navigate back to **/wae:wae/components/aggregators/aggregator/**<***network_name***>**/aggregator** and select an option under the **rule-set** drop-down list.

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 9.

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>
# 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: Topology Collection Using the IGP Database, on page 4, Configure Multilayer Collection Using the EPN-M Agent, and Configure Multilayer Collection Using the CTC Agent.

```
# show running-config networks network nimo
```

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    bgl_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
# commit
```

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

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

A status message will appear once collection is complete. Once it completes, check to see that the nodes are populated.

```
# show running-config networks network l3-network model nodes node
```

You can also check to see that the final model is also populated with L3 information.

```
# show running-config networks network l1-l3-final-model model nodes node
```

Run the L1 network collection.

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

You can check again to see that the final model is now populated with L1 information.

```
# show running-config networks network l1-l3-final-model model nodes node
```

You can also open WAE Design to view the final network model (**File** > **Open from** > **WAE Automation Server** and select the final network model.

Autonomous System (AS) Model Consolidation

The as-merger NIMO resolves interfaces, circuits, etc., that are shared between AS models to create a single, consolidated network model.

Before you begin

- Confirm that collection has been completed on the individual AS network models that you want to merge.

Note BGP collection must be part of all AS models.

- Any AS network models that use the topo-bgpls-xtc NIMO must each have an Autonomous System Number (ASN) assigned to it. For more information, see Topology Collection Using XTC, on page 5.

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

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

Step 3 Click **Add**.

Step 4 Click the **nimo** tab.

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

Step 6 Click **as-merger-nimo**.

Step 7 From **sources**, add the individual AS models you want to merge.

Step 8 From **subtrees**, enter **model** as the value.

Step 9 From **generate capabilities**, select one of the following:

- **false**—Select this option if the AS sources are DARE networks.
- **true**—Select this option if the AS sources are not DARE networks and are created by other NIMOs.

Step 10 Click the **Commit** button.

Step 11 Click **merge**.

Example

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

```
# config
# networks network <as-merger-name> nimo as-merger sources [ <AS1-source> <AS2-sourse>
<ASn-source> ]
# networks network <as-merger-name> nimo as-merger subtrees [ model ]
# networks network <as-merger-name> nimo as-merger generate-capabilities <flag>
# commit
```

For example:

```
# config
# networks network AS100AS200 nimo as-merger sources [ AS6100 AS6200 ]
# networks network AS100AS200 nimo as-merger subtrees [ model ]
# networks network AS100AS200 nimo as-merger generate-capabilities <false>
# commit
```

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 4 or Topology Collection Using XTC, on page 5.
- Telemetry must be configured. See the Configure Telemetry in WAE topic.

Note You cannot use the Cisco WAE UI to configure this collection.

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 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.

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 Create a Network Model.

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 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

Note This is a Beta feature and has not been thoroughly tested.

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 Basic Topology Collection, on page 4.

Step 1 Copy the <*wae_installation_directory*>/packages/experimental/cisco-wae-lsp-config-nimo and <*wae_installation_directory*>/packages/experimental/cisco-wae-nso-agent packages to the <*wae_run_directory*>/packages directory.

```
# cp -R <wae_installation_directory>/packages/experimental/cisco-wae-lsp-config-nimo
<wae_run_directory>/packages
# cp -R <wae_installation_directory>/packages/experimental/cisco-wae-nso-agent packages
<wae_run_directory>/packages
```

Step 2 Stop WAE and reload the package.

```
# wae --stop
# wae --with-package-reload
```

Step 3 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 IP address—The NSO instance IP address.
- 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 4 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 5 From the Expert Mode, navigate to **/wae:networks**.

Step 6 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 7 Click **Add**.

Step 8 Click the **nimo** tab.

Step 9 From the **Choice - nimo-type** drop-down list, choose **lsp-config-nimo**.

Step 10 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 11 Click the **Commit** button.

Step 12 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 5.

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 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.

Step 9 Click the **Commit** button.

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

Step 11 To verify that the collection ran successfully, navigate to back to the network (**/wae:networks/network/<*network-name*>**) and click the **model** tab.

Step 12 Click **nodes**. A list of nodes and details appears, indicating a successful collection.

Step 13 Choose one of the nodes that you know has an LSP and click the **lsps** tab.

Step 14 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 following configuration parsing NIMOs:

- port-cfg-parse-nimo—Discovers LAG ports and Link Management Protocol (LMP) interfaces.
- srlg-cfg-parse-nimo—Discovers Shared Risk Link Groups (SRLG) configuration.
- vpn-cfg-parse-nimo—Discovers VPN configuration.
- lsp-cfg-parse-nimo—Discovers LSP configuration.

Before you begin

- A topology network model must exist. See Create a Network Model.
- The Configuration Parsing agent must be configured and running. For more information, see Configure the Configuration Parsing Agent Using the Expert Mode.

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 contains the source network and NIMO names; for example, networkABC_port-cfg-parse.

Step 3 Click **Add**.

Step 4 Click the **nimo** tab.

Step 5 Choose the applicable NIMO as the NIMO type.

Step 6 Click NIMO link and enter the following information:

- **source-network**—Choose the applicable network model that contains topology information.
- **cfg-parse-agent**—Choose a configuration parsing agent.
- **connect-timeout**—Enter a timeout in minutes.
- **lag**—Advanced option for the port-cfg-parse-nimo. Enable LAG discovery of port members.
- **lmp**—Advanced option for the port-cfg-parse-nimo. Enable discovery of LMP interfaces.

Step 7 Click the **Commit** button.

Step 8 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 1.

Port Config Parse Advanced Options

This topic describes advanced options available when creating the port-cfg-parse NIMO.

Option	Description
parse	Choose config objects to parse. Select from: • lmp—Enable discovery of LMP interfaces. • lag—Enable LAG discovery of port members. • lmp and lag—Enable discovery of LMP interfaces and LAG discovery of port members. • topology—Enable discovery of topology configurations.
connect-timeout	Enter a timeout in minutes.

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.

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 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 BGP Topology Advanced Options, on page 17.

Step 7 Click the **peer-protocol** tab and enter applicable IPv4 and IPv6 addresses.

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.
asn-include	Specify ASNs of interest. If used, peer discovery is restricted to this list. The default is to peer with all discovered external ASNs.
find-internal-asn-links	Find links between two or more internal ASNs. Normally this action is not required because IGP discovers these links.

Option	Description
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 Basic Topology Collection, on page 4.

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 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 (blacklists) 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

As of WAE 7.1.2, the preferred method of inventory collection is to use the external-executable-nimo. See the CLI example at the end of this procedure that demonstrates using the external-executable-nimo.

Before you begin

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

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 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
```

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

1. Generate auth.enc:

```
# 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
```

2. Configure external-executable-nimo:

```
# networks network <network_model_name_getinventory> nimo external-executable-nimo
# networks network <network_model_name_getinventory> nimo external-executable-nimo
source-network <source_network>
# networks network <network_model_name_getinventory> nimo external-executable-nimo
storage-format <native_or_YANG>
# networks network <network_model_name_getinventory> nimo external-executable-nimo advanced
```

```
argv "<wae_run_directory>/packages/cisco-wae-inventory-nimo/priv/bin/get_inventory -auth-file
 <directory>/auth.enc
-plan-file $$input -out-file $$output -net-access-file
<wae_run_directory>/packages/cisco-wae/priv/etc/net_access.txt
-login-allowed false"

# networks network <network_model_name_buildinventory> nimo external-executable-nimo
source-network <network_model_name_getinventory>
# networks network <network_model_name_buildinventory> nimo external-executable-nimo
storage-format <native>
# networks network <network_model_name_buildinventory> nimo external-executable-nimo advanced
argv "<wae_run_directory>/packages/cisco-wae-inventory-nimo/priv/bin/build_inventory
-plan-file $$input -out-file $$output"
# networks network <network_model_name_buildinventory> plan-archive archive-dir
<archive_directory>
source file
# commit
```

Note We recommend that the storage-format is set to "native" for faster performance. If the storage-format is set to "native", then source should be set to "file" as shown in the above example.

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.

- master 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 12.
- If collecting LSP traffic, an LSP network model must exist. See LSP Collection Using SNMP, on page 18.
- Maximum number of open files (ulimit -n): 1,000,000

Limitations

- Node traffic information from external interfaces is not collected.

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 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.

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 28 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 28 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 28 to tune the polling period.

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**.

What to do next

After completing traffic collection, you can create demands (Demand Mesh Creation, on page 31) and perform demand deduction (Demand Deduction, on page 32).

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.

Option	Description
log-file	Traffic poller log file.
net-record-file	Enter the filename where recorded SNMP messages are saved.
verbosity	Set the poller logging level. The default is 40. • 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), WMD will pull the traffic statistics from the RPC API. The traffic statistics will not be sent to CDB.
connect-timeout	Enter the maximum execution time for traffic population, in minutes.
storage-format	Select the YANG (stores in CDB) or native storage (stores in plan files) format. Performance is faster if native storage is selected.
kafka-config	
broker-url	URL of Kafka broker.
zookeeper-url	URL of Kafka zookeper.

Tuning Traffic Polling

To run traffic polling efficiently, do the following:

1. 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
```

2. View the poller.log file. By default, the file is located in `<wae_run_time_directory>/logs/<network_name>-poller.log`.
3. Search for and filter the following text:
   - Interface Traffic Poller: Collection complete. Duration:
   - LSP Traffic Poller: Collection complete. Duration:
4. Note the duration on average and during a worst-case scenario.

For example, interface polling takes 30 seconds on average and 45 seconds in a worst-case scenario. LSP polling takes 90 seconds on average and a maximum of 120 seconds. To run interface polling efficiently, we recommend starting with the worst-case (higher) numbers. In this example, start with the interface poller every 45 seconds and the LSP poller every 120 seconds. Then, as time passes, you can tune these numbers as needed.

To calculate traffic, you need at least two counters (two polling runs). Counters are chosen among those stored in memory using a sliding window. The advanced `raw-counter-ttl` option dictates how long the counters are kept. The advanced `stats-computing-minimum-window-length` and `stats-computing-maximum-window-length` options specify the size of the sliding window. The basic calculation on how to configure these parameters is:

```
stats-computing-minimum-window-length = ( poller duration ) * 5
stats-computing-maximum-window-length = stats-computing-minimum-window-length * 1.5
raw-counter-ttl = stats-computing-minimum-window-length * 3 / 60
```

In this example, because LSP collection takes longer than interface collection, use the LSP worst-case duration (120 seconds) to determine the sliding window size. You get the following numbers:

```
stats-computing-minimum-window-length = 120*5 = 600
stats-computing-maximum-window-length = stats-computing-minimum-window-length * 1.5 = 600
* 1.5 = 900
raw-counter-ttl = stats-computing-minimum-window-length * 3 / 60 = 30
```

You can run traffic calculation as often as every 30 to 45 seconds, or as slowly as every 120 seconds, as configured with the advanced `scheduler-interval` option. Set it to 120 seconds to conserve CPU. It might take a while to calculate traffic in large networks, so make sure to set an adequate **`connect-timeout`** option. You can find the actual period in `logs/ncs-java-vm.log`. For example:

```
Traffic calculation took (ms) 3750
```

You can tune these parameters more aggressively or conservatively. We recommend starting with more conservative settings, and then tuning them as needed. If you see traffic being dropped from the output, you can increase the `stats-computing-maximum-window-length` and `raw-counter-ttl` options accordingly.

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.

Note As of Cisco WAE 7.1.2, we recommend to use the external-executable-nimo for adding layout properties. See the CLI example at the end of this topic.

Before you begin

- A basic topology network model must exist. See Basic Topology Collection, on page 4.

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 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 a source network.

Note The source network cannot have a preexisting layout template assigned to it.

Step 7 Click the **Commit** button.

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

Step 9 Launch WAE Design and choose **File** > **Open From** > **WAE Automation Server**.

Step 10 Enter the appropriate details, choose the plan file for the network model you just created (networkABC_layout), and click **OK**.

Step 11 Edit the layout. See the "Using Layouts" chapter in the Cisco WAE Network Visualization Guide.

Step 12 Create and send the patch (**Tools** > **Patches** > **Create**). See the "Patch Files" chapter in the Cisco WAE Design User Guide.

Step 13 From the Expert Mode, navigate back to the layout-nimo network model (networkABC_layout).

Step 14 Click the **layouts** tab.

Step 15 Click **layout** to confirm that the table has been populated with layout data. The next time you open the plan file from WAE Design, the topology is displayed with the saved layout properties.

Example

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

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

4. Use the WAE CLI (config mode) to add the layout properties:

Note We recommend that the storage-format be native for faster performance.

```
# networks network <layout_network_model> nimo external-executable-nimo
# networks network <layout_network_model> nimo external-executable-nimo source-network
NetworkABC_demands
# networks network <layout_network_model> nimo external-executable-nimo storage-format
native
# networks network <layout_network_model> nimo external-executable-nimo advanced
argv "copy_from_template -plan-file $$input -out-file $$output -template-file
/home/centos/plan_files/template_01.pln -method visual -visualL1 true"
```

Demand Mesh Creation

Demand meshes are a time-efficient way of creating numerous demands for all or part of the network. The demandmesh-creator-nimo creates a demand mesh between a set of source and destination nodes. Use the resulting demand mesh network as the source network in the demand deduction NIMO.

Before you begin

A basic topology network model (Basic Topology Collection, on page 4) and traffic collection (Traffic Collection, on page 26) must exist. See .

Note As of Cisco WAE 7.1.2, we recommend to use the external-executable-nimo for demand creation. See the CLI example at the end of this topic.

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 contains the source network and NIMO names; for example, networkABC_demandmesh.

Step 3 Click **Add**.

Step 4 Click the **nimo** tab.

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

Step 6 Click **demandmesh-creator-nimo** and enter the traffic polling network.

Step 7 Click the **input** tab and enter:

- **source-nodes**—Enter nodes to restrict the demand mesh source to a specified set of nodes, external asynchronous systems, or external endpoints. You can view nodes from **/wae:networks/network/<*network_model*>/model/nodes**.
- **both-directories**—Include all demands from the destination to source and also from the source to destination. The default is true.
- **service-class**—Assign a service class type to demands that are created (for example, QoS). If empty, a default class is used.
- **topology**—Enter the topology. The default is that demands are applied to all topologies.

- **choice-destination: destination-nodes**—Choose this option if you want to create demands to destinations other than what has been selected as the source.
- **choice-destination: destination-equal-source**—If set to **true**, the destination node is set to source. The default is true.

Note To get a full demand mesh, do not edit any fields and set destination-equal-source to true. This picks up all the nodes in the network and creates all possible demands (including demands to the node itself).

Step 8 Click the **Commit** button.

Step 9 Navigate back to the **demandmesh-creator-nimo** tab and click **run** > **Invoke run**.

Step 10 To confirm that demands have been created, navigate to **/wae:networks/network/<*network_model*>/model/demands**. The table is populated with demand information.

Example

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

Note We recommend that the storage-format be native for faster performance.

```
# networks network <demand_mesh_network_model> nimo external-executable-nimo
# networks network <demand_mesh_network_model> nimo external-executable-nimo source-network
 <traffic_poll_network>
# networks network <demand_mesh_network_model> nimo external-executable-nimo storage-format
 <native_or_YANG>
# networks network <demand_mesh_network_model> nimo external-executable-nimo advanced
argv "dmd_mesh_creator -plan-file $$input -out-file $$output -dest-equals-source true"
```

What to do next

After creating demands, you can perform demand deduction (Demand Deduction, on page 32).

Demand Deduction

Traffic can be measured on interfaces, interface queues, and LSPs. You can use demand deduction to estimate demand traffic based on any of these measurements. For information on demand deduction, see the Cisco WAE Design User Guide. The demand-deduction-nimo runs demand deduction using measured SNMP traffic and by applying demand mesh builds of end-to-end demands (traffic matrix) to the network model.

Before you begin

A consolidated network model (aggregator NIMO) with demands (for example, networkABC_demandmesh) and interface measurements (for example, networkABC_traffic_poller) must exist. Use the demand mesh network as the source network in the demand deduction NIMO.

Note As of Cisco WAE 7.1.2, we recommend to use the external-executable-nimo for demand deduction. See the CLI example at the end of this topic.

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 contains the source network and NIMO names; for example, networkABC_demand_deduction.

Step 3 Click **Add**.

Step 4 Click the **nimo** tab.

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

Step 6 Click **demand-deduction-nimo** and enter the source network. The source network is typically a consolidated network model (aggregator NIMO) that has demands and traffic poller information.

Step 7 Click the **input** tab.

- **nodes**—Use measured traffic on nodes. The default value is true.
- **interfaces**—Use measured traffic on interfaces. The default value is true.
- **lsps**—Use measured traffic on LSPs.
- **remove-zero-bw-demands**—Remove any demands with no (zero) traffic (or less than the `zero-bw-tolerance` option). The default is true.
- **zero-bw-demands-tolerance**—Enter a tolerance value (less than zero) that will be considered as zero traffic.
- **demand-upper-bound**—Enter an upper bound on the demand traffic levels. A warning is issued if the upper bound is reached. The default is 10,000 Mb/s.

Step 8 Click the **Commit** button.

Step 9 Navigate back to the **demand-deduction-nimo** tab and click **run** > **Invoke run**.

Step 10 To confirm that demand deduction succeeded, navigate to **/wae:networks/network/<*network_model*>/model/demands** and see if the traffic column was populated.

Example

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

Note We recommend that the storage-format be native for faster performance.

```
# networks network <demand_ded_network_model> nimo external-executable-nimo
# networks network <demand_ded_network_model> nimo external-executable-nimo source-network
 <demand_mesh_network>
# networks network <demand_network_model> nimo external-executable-nimo storage-format
<native_or_YANG>
# networks network <demand_ded_network_model> nimo external-executable-nimo advanced
argv "dmd_deduct -plan-file $$input -out-file $$output"
```

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.

As of Cisco WAE 7.1.2, we recommend to use this NIMO for inventory collection, applying layout information, creating demands, and demand deduction. For more information, see the following topics:

- Configure Inventory Collection, on page 23
- Demand Mesh Creation, on page 31
- Demand Deduction, on page 32
- Network Model Layout (Visualization), on page 29

Another example is documented in the Running External Scripts Example, on page 35 topic.

Before you begin

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

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_my_script.

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 From the storage-format drop-down list, select **yang** (stores information in CDB) or **native** (stores information in plan files). Performance is faster if native storage is selected.

Step 7 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. The default is 7.0.
- **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 $$input for the source network model and $$output for the resulting network model (after the script runs). 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 35.

Step 8 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 23 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/python /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:

```
admin@wae# show running-config networks network net_dest model nodes node cr1.atl interfaces
 interface to_cr1.hst description
networks network net_dest
 model nodes node cr1.atl
  interfaces interface to_cr1.hst
   description "My IGP metric is 37"
  !
 !
!
```