Embedded Collectors in single VM deployments

Embedded Collectors in Crosswork Network Controller

Embedded Collectors are included in the single VM deployment of Crosswork Network Controller. The solution collects network data via collector services and transfers it to Cisco Crosswork or external destinations using Kafka or gRPC. It is bundled with Cisco Crosswork Infrastructure and the Element Management Functions application as part of a unified package.

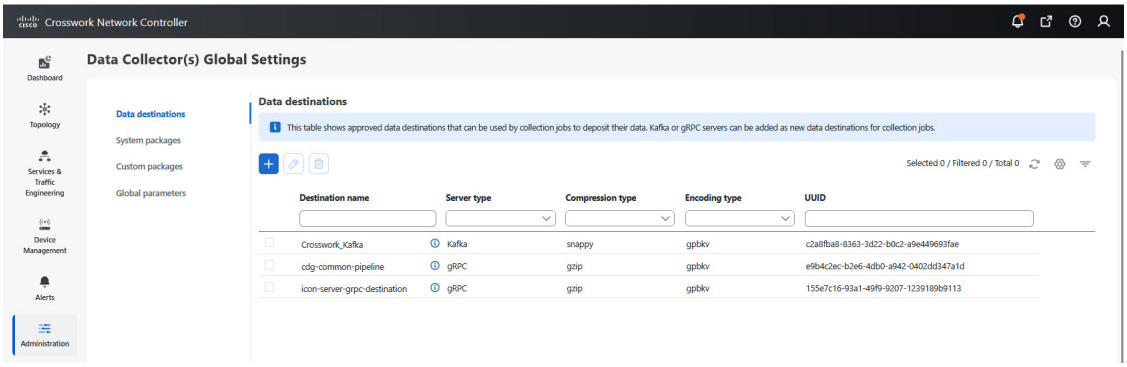
Key attributes:

- included with the bundled single VM deployment of Cisco Crosswork Network Controller
- collects network data from infrastructure devices using Embedded Collector services
- transfers data to Cisco Crosswork or external systems
- supports flexible data forwarding using Kafka or gRPC protocols, and
- enables streamlined integration with the overall Crosswork data and management ecosystem.

Access method for the Embedded Collector UI

You can access the Embedded Collectors through the management view from the **Data Collector(s) Global Settings** page in the Crosswork Network Controller UI.

Figure 1: Data Collector(s) Global settings



The Data Collector(s) Global Settings page allows you to perform the following administrative operations:

- **Data destinations**: After collecting the telemetry data, the collectors deposit it to an internal or external data destination. By default, `Crosswork_Kafka` is an internal data destination. You can define the external destinations using the Cisco Crosswork UI or APIs.

  To send data to external destinations using collection jobs, you must have an additional license. Make sure that the appropriate license is activated before configuring collection jobs. For the licensing details, see Licensing requirements for external collection jobs, on page 4.

- **Device packages**: By using device packages, the collectors can extend the data collection capabilities for both Cisco applications and third-party devices. The collectors support system and custom packages.
  - **System packages**: The system device package includes several installation files that are delivered via an application-specific manifest file. Usually, the manifest file is in JSON format.
  - **Custom packages**: The collectors user interface enables you to configure the port numbers of the collector pods. These port numbers affect the data collection services.
  - **Global parameters**: The collectors user interface allows enables you to configure the port numbers of the collector pods, which affect the data collection services. From this window, you can also enable the resync operation that automatically syncs the USM details whenever change occurs.

Configuring data collections in Embedded Collectors

Before setting up the Embedded Collectors, you must understand how Crosswork is set up. For more information, see *Tasks to complete for initial setup* section in the *Set Up Crosswork Network Controller* chapter.

The tasks in this section are listed according to the default configuration that Crosswork supports for Cisco devices. Optional tasks are only required if you wish to use the advanced features.

Summary

This process explains how to configure Embedded Collectors for collecting and transmitting data to Cisco Crosswork and other applications. It describes the setup tasks required for basic operation and outlines optional configurations to extend data collection capabilities.

Workflow

The data collection process involves these stages:

1. Complete the Single VM deployment: This package bundles Cisco Crosswork Infrastructure, Embedded Collectors, and Element Management Functions. This integration eliminates the need for separate installation. For more information, see the *Install Cisco Crosswork Network Controller on a Single VM* chapter in the *Cisco Crosswork Network Controller 7.2 Installation Guide*.
2. (Optional) Extend Embedded Collectors capabilities:

   This stage describes optional configurations that extend the Embedded Collector capabilities.

Table 1: Setting up data collection in Embedded Collectors

When the user wants to....	Then refer to the steps in...
1. Verify that default collection jobs are running.	Check the health status of Embedded Collectors , on page 69
2. (Optional) Add device packages to support additional or third-party devices	Device packages, on page 15
3. (Optional) Configure external data destinations	Data destinations in Embedded Collectors, on page 3
4. (Optional) Create custom collection jobs as needed	Collection jobs and supported protocols , on page 24

Result

When the process is complete, Embedded Collectors transmit collected data to the Crosswork Network Controller and other configured applications. You can enable advanced features to extend data collection capability.

Data destinations in Embedded Collectors

A data destination in the context of Embedded Collectors is a component in the Crosswork infrastructure that

- serves as the endpoint where collected network telemetry and performance data from Embedded Collectors is sent for processing, storage, and analysis,
- in a single VM deployment, is typically the local Crosswork data processing engine co-located within the same VM along with Embedded Collectors, forwarding data through internal buses or message brokers to processing services, databases, and analytics engines, and
- equires configuration with accurate IP addresses/hostnames, port numbers, authentication credentials, and protocol specifications (such as gRPC, TCP, UDP), while respecting system resource constraints and network connectivity rules to ensure optimized performance and reliable data flow.

Licensing requirements for external collection jobs

To set up collection jobs that send data to the external destinations, you need extra license. We recommend installing the license before configuring Crosswork to use an external destination. If you don't install the license first, you can still use the feature for 90 days under the trial license before it gets disabled.

When the License Authorization Status is "Out of Compliance", Crosswork Network Controller continues to allow users to create new external collection jobs and to view or delete existing jobs. This state occurs if registration with Cisco Smart Software Manager is not completed after the evaluation period or if the device limit for external collection jobs is exceeded.

View the license status

Check whether your system is properly registered and authorized to use licensed features.

Use this task when you need to confirm that your system has an active license, is in compliance, and is enabled for reserved features. This ensures ongoing access and compliance with Cisco Smart Software Manager requirements.

Procedure

Step 1 Go to **Administration > Smart Licenses**.

The **Smart licenses** tab under the **Application management** page is displayed.

Step 2 Ensure that the status is as:

- Registration Status: Registered. Indicates you have registered with Cisco Smart Software Manager (CSSM) and are authorized to use the reserved licensed features.
- License Authorization Status: Authorized (In Compliance). Indicates you have not exceeded the device count in the external collection jobs.
- Under Smart Licensing Usage, the entry "CNC Collection RTM - External Application End-Point" should show a status of In Compliance.

Managing data destinations

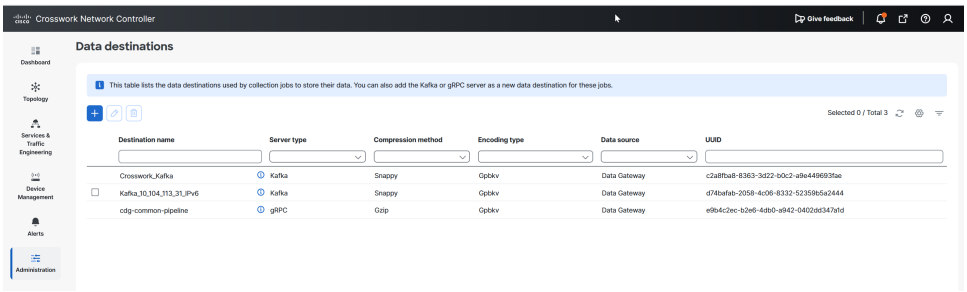
Explain how to manage external data destinations in Cisco Crosswork, including creating, modifying, and selecting data destinations for telemetry collection jobs.

Cisco Crosswork enables the creation of external data destinations, such as Kafka or external gRPC, which are utilized by the collection jobs to deposit the telemetry data.

To manage the data destinations, you can navigate to **Administration > Data Destinations**. From there, you have the options to

- add or modify a data destination
- delete any unused destinations, and
- view all the configured destinations.

Figure 2: Data destination



UUIDs for data destinations

The UUID is the unique identifier for the data destination. Cisco Crosswork automatically generates this ID when you create an external data destination.

When you create collection jobs using the Cisco Crosswork UI, you select the data destination from a drop-down list of configured destinations. When using the API, you need to know the UUID of the destination where the collector sends its data.

Add or edit a data destination

Add a new data destination or modify an existing one, enabling Embedded Collectors to send collected data to the desired destination.

Use this procedure to direct collected data to a new location or to update the parameters for an existing data destination, such as Kafka or gRPC endpoints.

Before you begin

Review the prerequisites and ensure you have all required information, such as destination details and authentication requirements.

Procedure

Step 1 Go to **Administration** > **Data Destinations**.

Step 2 On the **Data destinations** page:

a) To add a new destination, click **+ Add another**. Repeat this step for each additional collector. The **Data destinations** page opens.

b) To edit an existing destination, select it and click . The **Edit destination** page opens showing the current parameters. Update them as needed.

Note
When you update a data destination, the collector using it establishes a new session with that destination. Data collection pauses and resumes once the session is restored.

Step 3 Enter or update the required values for your external destination. If you are unsure about a value, use the default settings. For parameter information, see Parameters for adding and editing data destinations, on page 9.

Step 4 If you selected **Data Gateway** or **Any** as the data source and the server is set to **Kafka**, you can configure custom values for individual collectors when needed. To override the global properties for a Kafka destination, use the settings in the **Destination – Per Collector Properties** pane:

a) Select **Collector**.

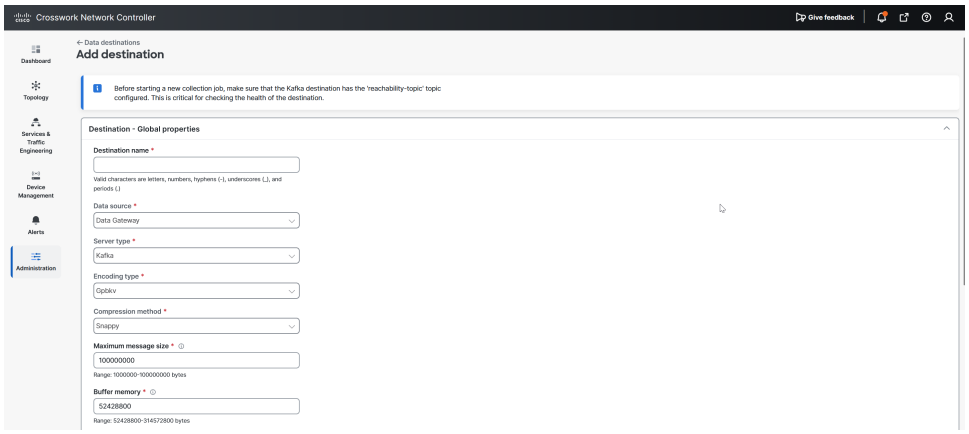
b) Enter values for all required fields.

- **Custom buffer memory**
- **Custom batch size**

  Note
  The **Custom batch size** cannot exceed the value of the **Custom buffer memory** at run time. If you do not enter a value for **Custom buffer memory**, validation occurs against the global Buffer Memory value.

- **Custom linger time**
- **Custom request timeout**

Figure 3: Add destination



c) Click **+ Add Another** to repeat for additional collectors.

Note
Properties set here override the global settings for the selected collectors. If you leave a field blank, the global properties are used.

Step 5 Select the protocol and host details in the **Connection details** sections. The supported protocols are IPv4, IPv6, and FQDN. For connection parameters, see Parameters for adding and editing data destinations, on page 9.

Note
FQDN is supported only for Kafka destinations.

Step 6 Complete the fields in Connection Details according to your connectivity type. Ensure the values match those configured on the external Kafka or gRPC server.

Note
You can change port numbers only for user-defined destinations. The ports of system-created destinations cannot be modified.

Step 7 (Optional) Enable security configurations.

a) If the data source is set to Data Gateway, the **Enable secure communication** check box is displayed. To connect securely to a Kafka or gRPC-based data destination, select this check box. Then select the type of authentication process from the available options.

- **Mutual-Auth**: Authenticates external server and the Crosswork Data Gateway collector after the CA certificate, and Intermediate certificate or Key is uploaded to the Crosswork UI. **Mutual-Auth** is the default authentication process.
- **Server-Auth**: Authenticates external server and the Crosswork Data Gateway collector after the CA certificate is uploaded to the Crosswork UI.

b) If the data source is set to Any or Application, the **Enable secure communication with mutual auth** check box is displayed. Select this check box to enable the security feature.

Step 8 For Kafka or gRPC-based destinations, select an authentication option:

- **Mutual-authentication**: Authenticates both server and Embedded Collectors after uploading CA certificate and Intermediate certificate or Key. (Default)
- **Server-authentication**: Authenticates both external server and Embedded Collectors using the CA certificate only.

Note
Authentication options are available only when **Enable secure communication** is enabled.

Step 9 Click **Save**.

The new or updated data destination is saved. The Embedded Collectors send data using the new configuration.

What to do next

1. This step applies if you have selected the data source as Data Gateway or Any.

   Create the required Kafka topics:

   - Configure the Kafka destination with the `reachability-topic` before initiating a new collection job. This is required for health monitoring of the destination.
   - The topics must exist in the external Kafka at the time of data dispatch; otherwise, Crosswork logs may display an exception:

     ```
     destinationContext: topicmdt4
     org.apache.kafka.common.errors.UnknownTopicOrPartitionException: This server does
     not host this topic-partition.
     ```

2. If you enabled secure communication, go to the **Certificate Management** page of your Crosswork UI and add the relevant certificate for the data destination. This step is required to establish secure communication.

Important When the data source is set to Data Gateway or Any, a missing or incomplete certificate causes the destination to enter an error state. The associated collection job is marked as Degraded. For details about certificate requirements and management, see your platform's certificate management documentation.

Requirements to add data destination

To use an external Kafka server as a data destination in Embedded Collectors, ensure these requirements are met:

- Determine the data source for your destination as Data Gateway or application (Element Management Function, Service Health, and so on). If you are unsure, you can select **Any**. The form shows or hides specific fields depending on the selected data source. For example, encoding types and security details. Be prepared to provide the fields that apply to your chosen source.
- Configure Kafka server properties: Set these properties on your external Kafka server:
  - `num.io.threads = 8`
  - `num.network.threads = 3`
  - `message.max.bytes= 30000000`

  Refer to Kafka documentation for detailed descriptions and usage of these properties.
- Configure data destinations
  - You can configure multiple data destinations as needed.
  - If you reinstall an existing external Kafka destination using the same IP address, restart the collectors for changes to take effect.
- Validate connectivity and custom properties
  - Verify port connectivity for each data destination. If the port is unreachable, the collection will fail.
  - Embedded Collectors support custom values in Kafka destination properties. This customization feature is not supported for gRPC destinations.
  - Global properties entered in the Destination Details pane are mandatory and applied to all Kafka destinations by default unless overridden at the individual collector level.
- Secure communication (Optional)

  You can secure communication between Crosswork and the Kafka data destination (either Crosswork Kafka or external Kafka). Note that encryption can impact performance.

  If TLS is required:
  - Keep the public certificate ready for TLS verification.
  - For client authentication, keep the client certificate and key files available.
  - If the key file is password-protected, configure the password as part of destination provisioning.
  - Embedded Collectors currently support IP-based certificates only.

    Ensure that:
    - Certificates are PEM-encoded.
    - Key files are in PKCS#8 format when generated using a Certificate Authority.

Parameters for adding and editing data destinations

These tables list and describe the parameters required for adding or editing the data destinations.

These parameters are grouped in two categories:

- General configuration parameters: Core options that affect the identity and encoding of the data destination.
- Connection configurations: Addressing and connectivity options, such as IP addresses or hostnames and ports.

Table 2: General configuration parameters for data destinations

Field	Description	Available in gRPC	Available in Kafka
Destination name	Enter a descriptive name (up to 128 characters). Valid characters include letters, numbers, hyphens (-), underscores (_), and periods (.). Avoid all other special characters.	Yes	Yes
Server type	Select the gRPC or Kafka for your data destination from the drop-down.	Yes	Yes

Field	Description	Available in gRPC	Available in Kafka
Data source		Yes	Yes

Field	Description	Available in gRPC	Available in Kafka
	Identifies which Crosswork component or application will use the external Kafka or gRPC destination to send data. This field determines the available configuration options, validation rules, and security features for the destination. Select the data source as • **Data Gateway**: destination exclusively used by Crosswork Data Gateway for telemetry and network data collection. • **Application**: destination exclusively used by Crosswork applications such as application events, alerts, and notifications. • **Any**: destination can be shared by both **Data Gateway** and **Applications**. **Note** • If you do not choose the data source, it defaults to **Data Gateway**. • If you set the data source to **Any**, you cannot change it later. To select a different data source, delete the destination and create a new one. • When you change the data source from **Data Gateway** to **Any** during editing a destination, Crosswork automatically switches the authentication type to mutual authentication and displays a warning message. • Embedded collectors does not monitor the availability of Kafka destinations configured with the dispatch source as **Applications** (Dispatch Source="application"). If a destination becomes unreachable, applications such as Service Health fail to detect the issue or notify users, which can result in silent data loss. • When upgrading from Crosswork 7.1 or earlier, all destinations default to **Data Gateway**. • Destinations that have **Application** as		

Field	Description	Available in gRPC	Available in Kafka
	the data source are removed after the upgrade.		
Encoding type **Note** This field appears only when the data source is set to Data Gateway or Any.	Choose encoding as Json or Gpbkv.	Yes	Yes
Compression method	Choose the compression method. Default is `snappy`. • Kafka supports `snappy`, `gzip`, `zstd`, `none` (zstd: Kafka ≥2.0). • gRPC supports `snappy`, `gzip`, `deflate`.	Yes	Yes
Maximum message size	Enter the max message size in bytes. • Default Value: 100,000,000 (100 MB). • Min: 1,000,000 (1 MB) • Max: 100,000,000 (100 MB).	No	Yes
Buffer memory	Enter required buffer memory in bytes. • Default Value: 52428800 bytes/52.4288 MB • Min: 52428800 bytes/52.4288 MB • Max: 314572800 bytes/314.5728 MB	No	Yes
Batch size	Enter required batch size in bytes. • Default Value: 1048576 bytes/1.048576 MB • Min: 16384 bytes/16.38 KB • Max: 314572800 bytes/314572.8 KB	No	Yes

Field	Description	Available in gRPC	Available in Kafka
Linger time	Enter linger time in ms. • Default Value: 2000 ms • Min: 0 ms • Max: 5000 ms	No	Yes
Request timeout	Enter request timeout duration in seconds. • Default Value: 30 seconds • Min: 30 seconds • Max: 60 seconds	No	Yes

Note

- For fields with limits, abide by min/max/default values to prevent configuration errors.
- Some parameters are only available for Kafka destinations, not for gRPC (e.g., buffer memory, batch size).
- Use informative names for easier management if you have many data destinations.

Table 3: Connection configuration parameters for data destinations

Field	Description	Available in gRPC	Available in Kafka
IPv4	Enter IPv4 Address/Subnet Mask and Port. Add multiple IPv4 addresses via **+ Add another** • Subnet mask: 1–32 • Port: 1024–65535	Yes	Yes
IPv6	Enter IPv6 Address/Subnet Mask and Port. Add multiple IPv6 addresses via **+ Add another**. • Subnet mask: 1–128 • Port: 1024–65535 IPv6 subnet mask ranges from 1 to 128 and ports range from 1024 to 65535.	Yes	Yes
FQDN	Enter Host Name, Domain Name, and Port (1024–65535). Add multiple via **+ Add another**.	Yes	Yes

Note Additional details:

- For all connection fields, you can add multiple entries using the **+ Add another** option.
- Always verify firewall settings to allow configured ports.
- Supported port ranges apply consistently across IPv4, IPv6, and FQDN settings.

View the data destination details

Review the configuration and attributes of a selected data destination within the Data Gateway.

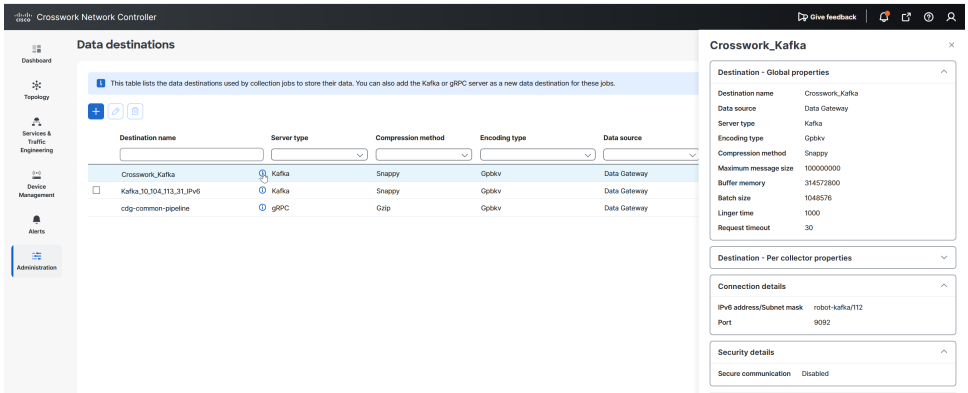
Use this task when you need to audit, verify, or analyze the data destination endpoints configured for your organization. Accurate review ensures that data flows to intended endpoints and assists in diagnosing configuration or routing issues.

Procedure

Step 1 Go to **Administration** > **Data Destinations**.

Step 2 Click the icon next to the data destination whose details you want to review.
Destination details appear with associated configuration information.

Figure 4: View destination details



The details for the selected data destination are displayed, allowing you to verify attributes and configuration.

What to do next

After viewing, confirm that the data destination matches the desired configuration. If you find any discrepancies, adjust the settings. If there are no discrepancies, continue regular monitoring.

Delete a data destination

Delete one or more data destinations from the system.

Data destinations store information collected by the data collector. You may need to delete a destination if it is no longer needed or to maintain system hygiene.

Before you begin

- A data destination can only be deleted if it is not associated with any collection job. Deleting a destination also removes all associated data subscriptions.
- Default destinations, such as `Crosswork_Kafka`, cannot be deleted.
- Check the **Collection Jobs** view to determine if any collection jobs are using the data destination.

Procedure

Step 1 Got to **Administration** > **Data Destinations**.

Step 2 Select one or more data destinations to delete from the displayed list.

Step 3 Click  .

Step 4 In the **Delete Data Destination(s)** pop-up, click **Delete** to confirm.

The selected data destinations are deleted from the system.

What to do next

Verify the destination has been removed.

Device packages

A device package is a data collection framework that

- defines communication protocols including commands, telemetry models, and SNMP objects required to retrieve device data from network devices
- translates device data into a standardized format for downstream processing within Embedded Collectors, and
- enables multi-vendor support allowing Embedded Collectors to communicate with multiple device types and vendors without requiring additional code changes.

Types of device packages

Embedded Collectors use device packages to define communication and data collection from network devices. These are categorized as system packages and custom device packages.

- System packages are preinstalled packages that ship with Crosswork. They include built-in definitions for Cisco and commonly supported third-party devices. These packages contain the standard CLI, SNMP, and telemetry collection models required for those devices.

  System device packages have these characteristics.

  - Installed automatically with Embedded Collectors.
  - Maintained and updated by Cisco.
  - Administrators cannot modify the system device packages. Only applications can modify these files. To modify the system device packages, contact the Cisco Customer Experience team.
  - Updated as part of Embedded Collectors or device package version upgrades.

  System packages ensure that Embedded Collectors can immediately begin collecting data from supported devices without requiring manual configuration.

- Custom device packages are provided by administrators. They extend the data collection capabilities of Embedded Collectors beyond the coverage of system packages. Typically, administrators use these to onboard unsupported devices, add vendor-specific MIBs, or introduce additional metrics.

  Custom device packages have these characteristics.

  - Created, imported, or obtained separately (for example, from Cisco or third-party vendors).
  - Uploaded manually through **Administration > Data Gateway > Device Packages**.
  - Can be edited, replaced, or deleted by administrators as needed.
  - Useful for integrating proprietary, non-Cisco, or newer device types.

Note Custom packages co-exist with system packages and take precedence when both define the same device type or data model.

Download system packages

Download system packages that are relevant for an application using the system's administration interface.

System packages are provided through an application-specific manifest in JSON format. They are centrally managed and updated whenever applications are installed or upgraded.

Before you begin

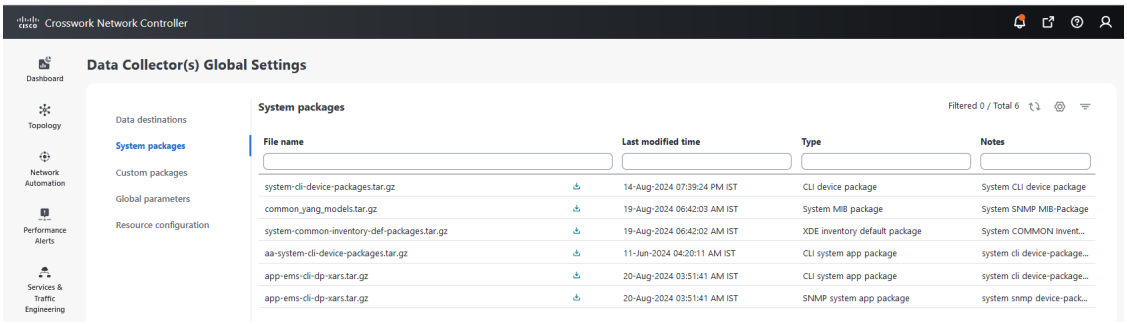
Identify the system package you need to download.

Procedure

Step 1 From the main menu, choose **Administration > Data Collector(s) Global Settings > System Packages**.

You will see the System Packages window.

Figure 5: System Packages Window



Step 2 Click the ⬇ button located next to the package name in the **File Name** column.

The selected system package is downloaded and available for use.

What to do next

Confirm the file has downloaded successfully.

Custom packages

A custom package is a bundle that

- enables user-supplied extensions for device communication and data collection
- allows integration of third-party device models and data formats, and
- supports enhanced monitoring and management within Crosswork Network Controller.

Types of custom packages

You can upload these types of custom packages to Crosswork Network Controller:

- CLI Device Package: Monitors device health for third-party devices using KPIs based on the CLI. All packages and their corresponding YANG models must be included in the `custom-cli-device-packages.tar.xz` file.
- Custom MIB Packages: Contains custom MIBs and device packages designed for third-party devices, and filter or format collected data for Cisco devices. You can edit these packages, which must be included in the `custom-mib-packages.tar.xz` file. The system supports only one custom MIB package.
- SNMP Device Package: Extends SNMP coverage via custom SNMP device packages in `.xar` format, using Embedded Collectors.
- Aggregate Package: Enables you to include multiple supported file extensions in one package. You can upload and download aggregate packages using the Crosswork UI.

Supported file extensions for aggregate packages

Aggregate packages may contain these file types:

- Collector file
  - YANG (.yang)
  - MIB (.mib, .my)
  - Definition (.def)
  - Device packages (.xar)
- Application files
  - Device-metadata (.yaml, .yml)
  - Zips (.zip)
  - SDU bundle (.sdu)

Requirements for proprietary MIBs

Proprietary MIBs are only needed when collection requests reference MIB TABLE names or SCALAR names from a proprietary MIB. If requests use OIDs, proprietary MIBs are not required. Standard MIBs are already included in the system for SNMP polling on third-party devices.

Upload the custom package

Add a new custom package to the system by uploading a prepared tar.gz bundle.

Custom packages allow administrators to introduce new capabilities or configurations by uploading bundled resources in a standard archive format. Supported package types include those for SNMP and CLI collectors.

Before you begin

Review the requirements and ensure that you meet the requirements.

Procedure

Step 1 From the main menu, go to **Administration > Data Collector(s) Global Settings > Custom packages**.

Step 2 On the **Custom packages** page, click  .

Step 3 In the **Add custom packages** window, select the appropriate package type from the **Type** dropdown.

Step 4 In the **File name** field, click to open the file browser, select your tar.gz custom package, and click **Open**.

Step 5 Add a description of the package in the **Notes** field. We recommend including a unique description for each package to easily distinguish between them.

Step 6 Click **Upload**.

The system processes your upload and adds the custom package to the selected collector or gateway. Confirmation is shown upon completion.

Requirements to upload custom packages

Ensure that all prerequisites are met and review the applicable upload restrictions before uploading custom packages. Observe these requirements:.

- MIB package dependencies: Ensure that the new MIBs include all necessary dependencies in the bundle to prevent import errors.
- Supported file extensions: The package must include only supported file types. For the complete list of supported extensions, see the referenced documentation.

  Supported file extensions: The package must include only supported file types. For the full list of supported extensions, refer to the relevant documentation. For a full list of supported extensions, see Custom packages, on page 17.
- Package format: Bundle all the package contents into a `.tar.gz` archive before uploading.
- Collector types: The archive must include at least one of the top-level directories:
  - `cli/`
  - `snmp/`
  - `common/`

  See Sample package directory structure, on page 20 for the recommended directory structure.
- To update a CLI package, click the Upload icon next to the filename on the Custom Packages page. This action replaces the existing file.
- When uploading multiple files, combine multiple `.xar` files into a single `.tar.gz` archive before uploading.

Performance considerations for uploading custom packages

Before uploading custom MIBs and YANGs to the Crosswork Network Controller, ensure that your packages are thoroughly performance-tested and optimized for your deployment scale.

- Optimize custom package efficiency to improve collection job performance.
- Test package performance prior to uploading to ensure compatibility and scalability for your environment.

For detailed validation instructions on custom MIBs and YANGs, refer to Cisco DevNet documentation.

Restrictions and validations for custom package uploads

Observe these restrictions and validations when uploading custom packages:

- Do not attempt to overwrite system MIBs with custom MIBs; this action is not supported and will result in a failed upload.
- Package archive requirements:
  - Include only the directories `cli/`, `snmp/`, and `common/` at the root level of the `.tar.gz` archive.
  - Avoid including parent folders or extra hierarchy levels to prevent exceptions during job execution.
- Crosswork validates only file extensions at upload time; file contents are not validated.

Sample package directory structure

This example shows a directory structure for an aggregate package:

```
├── cli
│   ├── defs
│   │   └── cli-def1.def
│   ├── device-metadata
│   │   ├── cli.yml
│   │   └── cli-device-metadata.yaml
│   ├── zips
│   │   └── cli-zip.zip
│   ├── sdus
│   │   └── cli-sdu.sdu
│   ├── xars
│   │   ├── cli-xar1.xar
│   │   └── cli-xar2.xar
│   └── yangs
│       ├── cli-yang1.yang
│       └── cli-yang2.yang
├── common
│   ├── defs
│   │   └── common-def1.def
│   ├── device-metadata
│   │   ├── common.yml
│   │   └── common-device-metadata.yaml
│   ├── zips
│   │   └── common-zip.zip
│   ├── mibs
│   │   ├── common-mib1.mib
│   │   └── common-mib2.my
│   ├── sdus
│   │   └── common-sdu.sdu
│   ├── xars
│   │   ├── common-xar1.xar
│   │   └── common-xar2.xar
│   └── yangs
│       ├── common-yang1.yang
│       └── common-yang2.yang
└── snmp
    ├── defs
    │   └── snmp-def1.def
    ├── device-metadata
    │   ├── snmp.yml
    │   └── snmp-device-metadata.yaml
    ├── mibs
    │   ├── snmp-mib1.mib
    │   └── snmp-mib2.my
    ├── sdus
    │   └── snmp-sdu.sdu
    ├── zips
    │   └── snmp-zip.zip
    ├── xars
    │   ├── snmp-xar1.xar
    │   └── snmp-xar2.xar
    └── yangs
        ├── snmp-yang1.yang
        └── snmp-yang2.yang
```

Delete the custom package

Remove a custom package from Cisco Crosswork and delete associated files.

Deleting a custom package automatically removes all YANG and XAR files associated with it and affects any collection tasks that rely on it.

Procedure

Step 1 From the main menu, choose **Administration > Data Collector(s) Global Settings > Custom Packages**.

Step 2 In the **Custom Packages** pane, select the package you want to delete.

Step 3 Click  .

Step 4 In the **Delete Custom Package** window that appears, click **Delete** to confirm.

Global collector parameters

Global settings define how the collectors operates and integrates with other network components such as devices, data destination. These configurations offer a centralized way to manage key operational parameters consistently across the Embedded Collectors.

The primary functions of global configuration options include:

- Controlling how Embedded Collectors connect to external systems. See Data destinations in Embedded Collectors, on page 3.
- Extending data collection capabilities to Cisco applications and third-party devices using device packages. See Device packages, on page 15.
- Setting default thresholds, ports, timeouts, and retry limits. See Configure the global parameters, on page 21.

Configure the global parameters

Guide users to configure global parameters for collectors in the network management system.

The Global Parameters window enables configuration of foundational operational settings for collectors, such as specifying data collection ports. Update these settings if your network uses non-standard ports to maintain proper collector integration and operation.

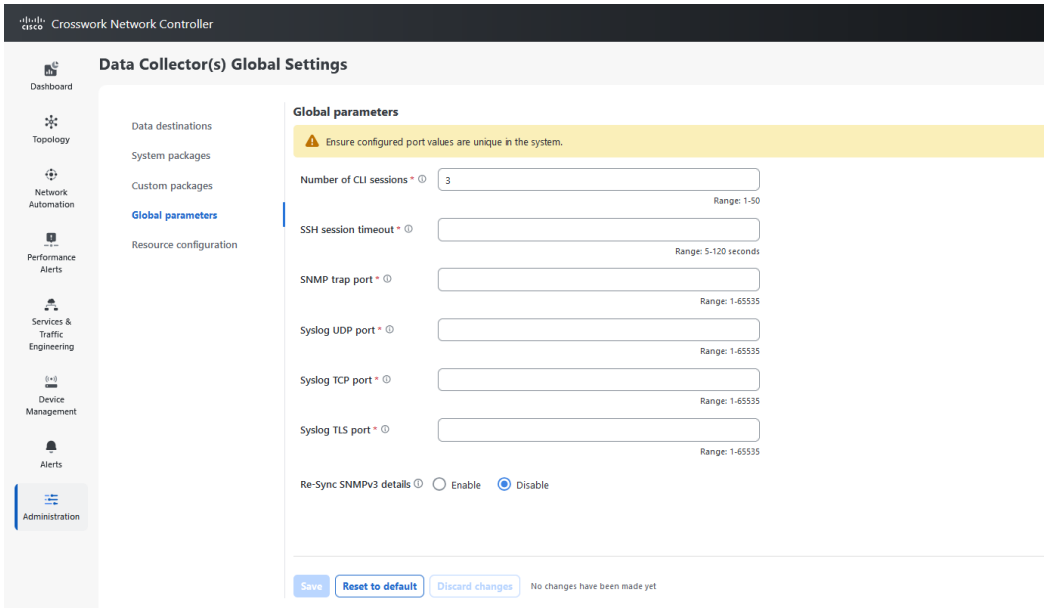
Before you begin

- Ensure port values you intend to update are valid and do not conflict with existing assignments.
- Confirm port numbers match those configured on your devices
- Ensure port values do not conflict with those on Embedded Collectors.
- Changes to Syslog and SNMP ports will apply to Embedded Collector instances running on Amazon EKS.

Procedure

Step 1 Navigate to **Administration > Data Collector(s) Global Settings > Global Parameters**.

Figure 6: Global parameters



Step 2 Update the parameter values as per your network requirement.

Step 3 If you are updating ports, select **Yes** in the **Global Parameters** window that appears to confirm that collectors can be restarted.

Note
Updating ports causes the collectors to restart and pause any collection jobs that are running.

Step 4 Click **Save**.

A confirmation message indicates whether the update succeeded or failed for Embedded Collectors.

What to do next

If any Embedded Collectors cannot be updated, an error window appears. During recovery, these collectors will automatically retry parameter updates; some may restart as part of this recovery process.

Note One reason global parameters might fail to update on an Embedded Collector is that the OAM channel is down. After the OAM channel is reestablished, Embedded Collectors attempt to send the parameters to the unsynchronized collectors, updating the values after comparing them with the existing ones.

Global parameters and descriptions

This section lists the global parameters and descriptions.

Note The port number you modify must match the numbers configured on your devices.

Table 4: Global parameters and descriptions

Parameter Name	Description	Default value for single VM deployment
Number of CLI sessions	Maximum number of CLI sessions that can be set up between an embedded data collector and devices. Change this value to increase or decrease the number of concurrent sessions allowed on the device, depending on requirements.	3 Accepted range is 1-50
SSH Session Timeout	The session timeout (in seconds) is the duration for which a CLI connection can remain idle in the CLI and SNMP collectors.	120 Accepted range is 5-900 seconds
SNMP Trap Port	Modify this value according to the deployment environment and configuration requirements.	31062 Accepted range is 30160–31560
Syslog UDP Port	Modify this value according to the deployment environment and configuration requirements.	30514 Accepted range is 30160–31560
Syslog TCP Port	-	30898 Accepted range is 30160–31560
Syslog TLS Port	-	30614 Accepted range of ports is 30160–31560

Parameter Name	Description	Default value for single VM deployment
Re-Sync SNMPv3 Details	USM details change whenever a device is rebooted or re-imaged. SNMPV3 collections stop working whenever there is a change in any of the USM details.	Disable By default, this option is disabled for security reasons. Automatic synchronization of updated USM (User Session Manager) information is not permitted to prevent unintended data collection with an incorrect source. When enabled, the system automatically updates USM information after changes, such as hardware updates or device reboots. This ensures that data collection continues without user intervention. If the option remains disabled, manual intervention is required to re-establish USM communication. This process can be performed by detaching and reattaching the device to the embedded collectors or toggling the device's admin state (Down then Up).

Collection jobs and supported protocols

A data collection process is a mechanism that

- enables applications to request network data through collection jobs
- allows the Crosswork Network Controller to assign these jobs to an appropriate collector, and
- ensures that the collector initiates and performs data collection based on the type of data requested.

The Embedded Collector collects data using supported protocols such as CLI, SNMP, gNMI (dial-in), and syslog. It can collect any type of data if it is able to forward it through one of these supported protocols. Embedded Collectors offer flexibility in data collection by supporting multiple protocols. This capability ensures compatibility with diverse data types and network devices.

Types of collection jobs

There are two types of data collection requests in Crosswork Network Controller:

1. A data collection request forwards data for internal processes within Cisco Crosswork. Cisco Crosswork creates system jobs for this purpose. If you want the Embedded Collectors to collect specific information from non-Cisco devices, you must use custom device packages. For more information about custom device packages, see Custom packages, on page 17.

   To learn how to build a model that enables Crosswork to communicate with non-Crosswork devices, see Cisco Devnet.

2. Data collection request to forward data to an external data destination. For more information about configuring the external data destinations (Kafka or gRPC), see Data destinations in Embedded Collectors, on page 3.

Supported types of collection jobs

- CLI-based collection
- SNMP-based collection
- Syslog-based collection
- gNMI-based collection

For each collection job you create, Embedded Collectors execute the collection request and forward the data to both internal and external destinations. A single collection request allows you to send the collected data to the Crosswork Network Controller and to an external data destination.

You can create collection jobs from the Cisco Crosswork Network Controller UI. You can also use APIs to create jobs, see Cisco DevNet. For example, you may create an SNMP-based collection job to regularly retrieve interface statistics from a device and deliver data to both the controller and an external monitoring server.

How collection job state transitions work

Collection jobs progress through distinct status stages, beginning with creation and ending with execution in the Crosswork system.

Summary

Collection jobs transition through distinct status stages as they move from creation to execution within the Crosswork system.

The key components involved in the process are:

- Collection job: the request that is created to collect data.
- Embedded Collector: the collector that validates and executes jobs.
- Crosswork Network Controller UI: the management interface where users monitor job status.

Workflow

The process involves these stages:

1. Initially, every collection job appears in the UI with the status Unknown.
2. The Embedded Collector receives the collection job and performs basic validation checks.
3. If the job passes validation, its status changes to Successful.
4. If the job fails validation, its status changes to Failed.

Result

Users can track and manage the lifecycle of collection jobs. They can quickly see whether their jobs have succeeded, failed, or are waiting to be processed.

Status changes in event-based collection jobs

These scenarios illustrate how status changes appear in event-based collection jobs.

- When data collection is successful, the status of the collection job changes from Unknown to Success in the Collection Jobs pane.
- When a device is detached from Embedded Collectors, all corresponding collection jobs are deleted, and the job status is displayed as Success in the Collection Jobs pane. No devices or collection tasks are displayed in the Job Details pane.
- When a device is attached to an Embedded Collector, Crosswork receives a new collection job with the status set to Unknown, which changes to Success after events are received from the device.
- If the device configuration is updated incorrectly on an already attached device (and Embedded Collectors has received the job and events), there is no change in the status of the collection task in the Jobs Details pane.
- If device inventory is updated with an incorrect device IP, the collection task status in the Jobs Details pane remains Unknown.

CLI collection jobs

A CLI collection job in the Embedded Collectors is a data collection method that

- uses command-line interface (CLI) commands to retrieve operational or configuration data from network devices,
- enables collection of real-time information when devices do not support other protocols such as SNMP or gNMI, and
- supports troubleshooting and monitoring by ensuring critical data is available.

Best practice for devices with banner configurations

If a banner configuration is currently enabled on your device, refer to the device's official documentation for instructions on how to disable it.

How CLI jobs collect data

Summary

CLI collection jobs automate data collection from devices. They reference designated destinations and device identifiers using the CLI protocol.

Workflow

The key steps for configuring a CLI collection job are:

1. Configure a data destination: To create a custom CLI collection job, configure a data destination. Each destination receives a unique UUID, which is required as destination_id in API payloads. You can create a destination using **Data Collectors > Data Destinations**.
2. Identify the device. The device uses a UUID instead of an IP address for identification.
3. For jobs built using the UI, Crosswork Network Controller automatically retrieves required UUIDs. For custom jobs, retrieve UUIDs manually.

Result

Setting up a CLI collection job with correctly referenced destinations and devices enables streamlined and accurate CLI-based data collection in the network management system.

Cadence for data collection

Cadence is a configuration parameter that

- determines how frequently the Embedded Collectors collect data from each device
- accepts a range between 10 and 604,800,000 milliseconds, and
- allows the user to tailor the data collection rate to operational needs.

For example, a cadence value of 60,000 milliseconds (1 minute) means data will be retrieved every minute. The minimum recommended cadence is 60 milliseconds. Select an appropriate cadence to balance data granularity with system and network performance.

Best practice for setting data collection cadence

- Set a minimum cadence of 60 milliseconds for most data collection jobs.
- Use a higher cadence (slower rate) for collecting consistent data, such as memory consumption or CPU utilization.
- Use a shorter cadence (faster rate) to collect data points that are more dynamic and fast-changing.
- High-frequency collection can increase the load on both devices and the Crosswork Network Controller. Consider this impact when choosing a cadence.
- Experiment with different cadence values to find the optimal balance between actionable insight and system performance.

Considerations for skipped collection attempts

When a collection attempt is skipped because a previous execution is still in progress, Embedded Collectors issue a warning log. However, the system does not generate an alert for this scenario. This behavior prevents overlapping data collection processes and helps maintain operational efficiency.

Sample payload of CLI collection job

A sample payload is a structured example that demonstrates how device data can be sent to an external system using a defined schema. It clarifies the usage of system-assigned identifiers such as UUIDs and provides a template for integrating Crosswork jobs with external Kafka destinations.

The sample payload includes the structure and values used when Crosswork sends device data to an external Kafka destination. It uses the UUID assigned by the Device Lifecycle Manager..

For detailed information about the API payload fields and usage examples, see the API documentation on Cisco Devnet.

```
{
  "collection_job": {
    "application_context": {
      "context_id": "collection-job1",
      "application_id": "APP1"
    },
    "collection_mode": {
      "lifetime_type": "APPLICATION_MANAGED",
      "collector_type": "CLI_COLLECTOR"
    },
    "job_device_set": {
      "device_set": {
        "devices": {
          "device_ids": [
            "658adb03-cc61-448d-972f-4fcec32cbfe8"
          ]
        }
      }
    },
    "sensor_input_configs": [
      {
        "sensor_data": {
          "cli_sensor": {
            "command": "show platform"
          }
        },
        "cadence_in_millisec": "60000"
      }
    ],
    "sensor_output_configs": [
     {
        "sensor_data": {
          "cli_sensor": {
            "command": "show platform"
          }
        },
        "destination": {
          "destination_id": "1e71f2fb-ea65-4242-8efa-e33cec71b369",
          "context_id": "topic1"
        }
      }
    ]
  }
}
```

SNMP collection jobs

An SNMP-based data collection job is a process that collects device data using SNMP through Embedded Collectors configured via the UI or API. It retrieves data based on the device's Management Information Base (MIB) and associated Object Identifiers (OIDs). This collection can be configured in two ways:

- MIB-based polling, which gathers data according to the device's supported MIB and OID definitions, and

- Trap-based listening, which collects SNMP traps by configuring the collector to listen for incoming trap messages.

Standard MIBs included with Crosswork enable collection of common device attributes. Custom or vendor-specific MIB packages can be uploaded for specialized devices.

Supported SNMP versions for polling and traps include SNMP v2 and SNMP v3. These versions offer various authentication protocols (such as HMAC_MD5 and HMAC_SHA variants) and privacy protocols (such as AES, DES, 3-DES, and Cisco-specific AES). This approach enables a flexible and secure collection of network device data for monitoring and management.

Supported SNMP versions and operations

Supported SNMP versions for data polling and traps are

Polling data

- SNMP V2
- SNMP V3 (no auth nopriv, auth no priv, authpriv)
- Supported auth protocols: HMAC_MD5, HMAC_SHA, HMAC_SHA2-512, HMAC_SHA2_384, HMAC_SHA2_256, and HMAC_SHA2_224
- Supported priv protocols: AES-128, AES-192, AES-256, CiscoAES192, CiscoAES256, DES, and 3-DES

Traps

- SNMP V2
- SNMP V3 (no auth nopriv, auth no priv, authpriv)

Sample SNMP device configuration commands

This table lists sample SNMP configuration commands for enabling polling and traps on V2 and V3 devices, including IP address, port notes, and authentication or naming requirements.

Table 5: Sample configuration to enable SNMP on device

Version	Command	Purpose
V2c	`snmp-server group <group_name> v2c` `snmp-server user <user_name> <group_name> v2c`	Defines the SNMP version, user or user group details.
	`snmp-server host <host_ip> traps SNMP version <community_string> udp-port 31062` `snmp-server host a.b.c.d traps version 2c v2test udp-port 31062`	Defines the destination to which trap data must be forwarded. **Note** The IP address must be the Data VIP address of the Embedded Collectors.
	`snmp-server traps snmp linkup` `snmp-server traps snmp linkdown`	Enables traps that notify about link status.

Version	Command	Purpose
V3 **Note** Password for a SNMPv3 user must be at least 8 bytes.	`snmp-server host <host_IP> traps version 3 priv <user_name> udp-port 31062`	Defines the destination to which trap data must be forwarded. **Note** The IP address must be the Data VIP address of the Embedded Collectors.
	`snmp-server user <user_name> <group_name> v3 auth md5 <password> priv aes 128 <password>`	Configures the SNMP server group and enables authentication for specified members in a named access list.
	`snmp-server view <user_name> < MIB > included`	Specifies the information that must be reported.
	`snmp-server group <group_name> v3 auth notify <user_name> read <user_name> write <user_name>`	Defines the SNMP version, user, or user group details.
	`snmp-server enable traps snmp [authentication ] [linkup ] [linkdown ] [warmstart ] [coldstart ]`	• When you use this command without any optional keywords, it enables authenticationFailure, linkUp, linkDown, warmStart, and coldStart traps. • When you include keywords with this command, it enables only the specified trap types. For instance, to enable only linkUp and linkDown SNMP traps for all interfaces, use the snmp-server enable traps snmp linkup linkdown command.

SNMP collector supported operations

- SCALAR
- TABLE
- WALK
- COLUMN

Notes for supported operations

- If a single collection requests for multiple scalar OIDs, you can pack multiple SNMP GET requests in a single `getbulkrequestquery` to the device.
- For TABLE operations, you can provide either a Table OID or a Column OID.
- There is an optional **deviceParams** attribute **snmpRequestTimeoutMillis** (not shown in the sample payloads) that should be used if the device response time is more than 1500 milliseconds.
  - Use **snmpRequestTimeoutMillis** unless you are certain that your device response time is high.
  - The value for **snmpRequestTimeoutMillis** should be specified in milliseconds:
  - The default and minimum value is 1500 milliseconds. There is no maximum value for this attribute.

SNMP collection job example

```
{
  "collection_job": {
    "application_context": {
      "context_id": "collection-job1",
      "application_id": "APP1"
    },
    "collection_mode": {
      "lifetime_type": "APPLICATION_MANAGED",
      "collector_type": "SNMP_COLLECTOR"
    },
    "job_device_set": {
      "device_set": {
        "devices": {
          "device_ids": [
            "c70fc034-0cbd-443f-ad3d-a30d4319f937",
            "8627c130-9127-4ed7-ace5-93d3b4321d5e",
            "c0067069-c8f6-4183-9e67-1f2e9bf56f58"
          ]
        }
      }
    },
    "sensor_input_configs": [
      {
        "sensor_data": {
          "snmp_sensor": {
            "snmp_mib": {
              "oid": "1.3.6.1.2.1.1.3.0",
              "snmp_operation": "SCALAR"
            }
          }
        },
        "cadence_in_millisec": "60000"
      },
      {
        "sensor_data": {
          "snmp_sensor": {
            "snmp_mib": {
              "oid": "1.3.6.1.2.1.31.1.1",
              "snmp_operation": "TABLE"
            }
          }
        },
        "cadence_in_millisec": "60000"
      }
```

```
    ],
    "sensor_output_configs": [
      {
        "sensor_data": {
          "snmp_sensor": {
            "snmp_mib": {
              "oid": "1.3.6.1.2.1.1.3.0",
              "snmp_operation": "SCALAR"
            }
          }
        },
        "destination": {
          "destination_id": "4c2ab662-2670-4b3c-b7d3-b94acba98c56",
          "context_id": "topic1_461cb8aa-a16a-44b8-b79f-c3daf3ea925f"
        }
      },
      {
        "sensor_data": {
          "snmp_sensor": {
            "snmp_mib": {
              "oid": "1.3.6.1.2.1.31.1.1",
              "snmp_operation": "TABLE"
            }
          }
        },
        "destination": {
          "destination_id": "4c2ab662-2670-4b3c-b7d3-b94acba98c56",
          "context_id": "topic2_e7ed6300-fc8c-47ee-8445-70e543057f8a"
        }
      }
    ]
  }
}
```

SNMP traps collection job

SNMP trap collection jobs are created only through the API. Trap listeners monitor specific ports and dispatch data to recipients according to their topics of interest.

How SNMP trap collection jobs work

When the Embedded Collector receives an SNMP trap, it performs these actions:

1. Checks if any collection job is created for the device.
2. Checks the trap version and community string.

Note To prevent Embedded Collectors from checking the community string for SNMP traps, select the **SNMP Disable Trap Check** check box when adding a device through the Crosswork Network Controller UI. For more information about this option, see *Add devices through the UI* in the *Cisco Crosswork Network Controller 7.1 Device Lifecycle Management* document.

3. For SNMP v3, the system also validates the user authentication protocol, privacy protocol, and credentials.

Note SNMPv3 authentication and privacy traps depend on the engineId of the device or router to maintain the local USM user tables. If the engineId changes, trap collection is interrupted. To restore trap reception, detach the respective device and then reattach it.

Best practice for enabling SNMP traps

- Before starting the SNMP trap collection, install the Common EMS Services application and configure the host information for SNMP.
- Embedded Collectors listen on UDP port 31062 for traps.
- Before submitting SNMP trap collection jobs, ensure that SNMP traps are properly configured on the device and directed to the Data VIP address of the Embedded Collector.

Types of SNMP trap filters

Embedded Collectors filter traps using the trap OID specified in the sensor path and send only the requested traps. The job can remain in the Unknown state in these scenarios:

- If the collection job is invalid, the status of the job remains "Unknown."
- If configuration is missing on the device, the status of the job remains "Unknown."
- If no trap is received, the status of the job remains "Unknown."

For a list of supported traps and MIBs, refer to *List of pre-loaded traps and MIBs for SNMP collection*.

Table 6: List of Supported Non-Yang/OID based Traps

Sensor path	Purpose
*	To get all the traps pushed from the device without any filter.
MIB level traps	OID of one MIB notification (Ex: 1.3.6.1.2.1.138.0 to get all the isis-mib level traps)
Specific trap	OID of the specific trap (Ex: 1.3.6.1.6.3.1.1.5.4 to get the linkUp trap)

Sample payload of SNMP collection job

In this example, Crosswork sends an SNMP trap collection job to receive SNMP traps from network devices.

For detailed information about the API payload fields and usage examples, see the API documentation on Cisco Devnet.

```
{
  "collection_job": {
    "application_context": {
      "context_id": "collection-job1",
      "application_id": "APP1"
    },
```

```
    "collection_mode": {
      "lifetime_type": "APPLICATION_MANAGED",
      "collector_type": "TRAP_COLLECTOR"
    },
    "job_device_set": {
      "device_set": {
        "devices": {
          "device_ids": [
            "a9b8f43d-130b-4866-a26a-4d0f9e07562a",
            "8c4431a0-f21d-452d-95a8-84323a19e0d6",
            "eaab2647-2351-40ae-bf94-6e4a3d79af3a"
          ]
        }
      }
    },
    "sensor_input_configs": [
      {
        "sensor_data": {
          "trap_sensor": {
            "path": "1.3.6.1.6.3.1.1.4"
          }
        },
        "cadence_in_millisec": "60000"
      }
    ],
    "sensor_output_configs": [
      {
        "sensor_data": {
          "trap_sensor": {
            "path": "1.3.6.1.6.3.1.1.4"
          }
        },
        "destination": {
          "destination_id": "4c2ab662-2670-4b3c-b7d3-b94acba98c56",
          "context_id": "topic1_696600ae-80ee-4a02-96cb-3a01a2415324"
        }
      }
    ]
  }
}
```

Enabling trap forwarding with OID identification

To identify the type of trap from the data received at the destination, look for the *oid* (OBJECT_IDENTIFIER, for example, `1.3.6.1.6.3.1.1.4.1.0`) and *strValue* associated to the *oid* in the OidRecords (application can match the OID of interest to determine the kind of trap).

These are the sample values used in the payload to forward traps to external applications:

- Link up

  ```
  1.3.6.1.6.3.1.1.4.1.0 = 1.3.6.1.6.3.1.1.5.4
  ```

- Link down

  ```
  1.3.6.1.6.3.1.1.4.1.0 = 1.3.6.1.6.3.1.1.5.3
  ```

- Syslog

  ```
  1.3.6.1.6.3.1.1.4.1.0 = 1.3.6.1.4.1.9.9.41.2.0.1
  ```

- Cold start

  ```
  1.3.6.1.6.3.1.1.4.1.0 = 1.3.6.1.6.3.1.1.5.1
  ```

Sample payload to forward traps to external applications

```
{
  "nodeIdStr": "BF5-XRV9K1.tr3.es",
  "nodeIdUuid": "C9tZ5lJoSJKf5OZ67+U5JQ==",
  "collectionId": "133",
  "collectionStartTime": "1580931985267",
  "msgTimestamp": "1580931985267",
  "dataGpbkv": [
    {
      "timestamp": "1580931985267",
      "name": "trapsensor.path",
      "snmpTrap": {
        "version": "V2c",
        "pduType": "TRAP",
        "v2v3Data": {
          "agentAddress": "172.70.39.227",
          "oidRecords": [
            {
              "oid": "1.3.6.1.2.1.1.3.0",
              "strValue": "7 days, 2:15:17.02"
            },
            {
              "oid": "1.3.6.1.6.3.1.1.4.1.0",  // This oid is the Object Identifier.
              "strValue": "1.3.6.1.6.3.1.1.5.3" // This is the value that determines the
kind of trap.
            },
            {
              "oid": "1.3.6.1.2.1.2.2.1.1.8",
              "strValue": "8"
            },
            {
              "oid": "1.3.6.1.2.1.2.2.1.2.8",
              "strValue": "GigabitEthernet0/0/0/2"
            },
            {
              "oid": "1.3.6.1.2.1.2.2.1.3.8",
              "strValue": "6"
            },
            {
              "oid": "1.3.6.1.4.1.9.9.276.1.1.2.1.3.8",
              "strValue": "down"
            }
          ]
        }
      }
    }
  ],
  "collectionEndTime": "1580931985267",
  "collectorUuid": "YmNjZjEzMTktZjFlOS00NTE5LWI4OTgtY2Y1ZmQxZDFjNWExOlRSQVBfQ09MTEVDVE9S",
  "status": {
    "status": "SUCCESS"
  },
  "modelData": {},
  "sensorData": {
    "trapSensor": {
      "path": "1.3.6.1.6.3.1.1.5.4"
    }
  },
  "applicationContexts": [
    {
      "applicationId": "APP1",
      "contextId": "collection-job-snmp-traps"
    }
```

```
    ]
}
```

Syslog collection jobs

A Syslog collection job is a data collection process that

- uses Embedded Collectors to gather Syslog-based events from network devices in RFC 5424 and RFC 3164 formats
- employs SyslogSensors to filter events based on severity, facility, and regular expressions, and
- applies logical operators to customize filtering, reducing noise and optimizing the volume of collected Syslog data.

Supported Syslog formats

Embedded Collectors support the collection of Syslog-based events from network devices. The collectors support these Syslog message formats:

- RFC 5424
- RFC 3164

The supported Syslog formats are:

- RFC5424 Syslog format
- RFC3164 Syslog format

Filtering the Syslog events

A Syslog event filter is a configuration mechanism that:

- manages and controls the volume of Syslog data collected from devices through SyslogSensors
- supports PRI-based and filter-based rules that help capture relevant Syslog events for network monitoring and analysis, and
- applies filters based on severity, facility, or regular expressions to forward only required events, thereby reducing noise, optimizing storage, and streamlining downstream processing.

Syslog filters allow the use of logical operators such as `AND` and `OR` to define up to three filter combinations. This approach provides flexibility in how filters are evaluated.

Configure syslog data collection for Embedded Collectors

Enable syslog data collection from network devices using Embedded Collectors.

Use this procedure to configure the Embedded Collectors to receive syslog event data and forward it for monitoring and analysis with Crosswork.

Before you begin

Confirm the devices to be monitored are reachable and support the necessary capabilities.

Use these steps to configure syslog data collection.

Procedure

Step 1 Install the Element Management Functions application and configure the host information for syslog. For additional details, refer to the *Cisco Crosswork Network Controller 7.2 Installation Guide*.

Step 2 Add the device and select the `YANG_CLI` capability.

Step 3 Configure the required parameters to enable syslog data collection from Embedded Collectors.

Note
The order of the steps does not affect the outcome. However, steps 2 and 3 are required; skipping either will prevent syslog data collection.

Additional information:

- For example configurations, refer to
  - Sample syslog configuration for single VM deployment
  - Sample device configuration for single VM deployment
- Review your platform-specific documentation to obtain configuration guidance.

Syslog data from selected devices is collected and made available in Crosswork via the Embedded Collectors.

Sample syslog collection payload

In this example, Crosswork sends a syslog-trap collection job to receive syslog messages sent from network devices. For detailed information about the API payload fields and usage examples, see the API documentation on Cisco Devnet.

```
{
  "collection_job": {
      "job_device_set": {
      "device_set": {
        "devices": {
          "device_ids": [
            "c6f25a33-92e6-468a-ba0d-15490f1ce787"
          ]
        }
      }
    },
    "sensor_output_configs": [
      {
        "sensor_data": {
          "syslog_sensor": {
            "pris": {
                "facilities": [0, 1, 3, 23,4],
                "severities": [0, 4, 5, 6, 7]
            }
        }
        },
        "destination": {
          "context_id": "syslogtopic",
          "destination_id": "c2a8fba8-8363-3d22-b0c2-a9e449693fae"
        }
```

```
      }
    ],
    "sensor_input_configs": [
      {
        "sensor_data": {
          "syslog_sensor": {
            "pris": {
                "facilities": [0,1, 3, 23,4],
                "severities": [0,4, 5, 6, 7]
            }
        }
        },
        "cadence_in_millisec": "60000"
      }
    ],
    "application_context": {
      "context_id": "demomilesstone2syslog",
      "application_id": "SyslogDemo2"
    },
    "collection_mode": {
      "lifetime_type": "APPLICATION_MANAGED",
      "collector_type": "SYSLOG_COLLECTOR"
    }
  }
}
```

Syslog collection jobs

A Syslog collection job is a data collection process that

- uses Embedded Collectors to gather Syslog-based events from network devices in RFC 5424 and RFC 3164 formats
- employs SyslogSensors to filter events based on severity, facility, and regular expressions, and
- applies logical operators to customize filtering, reducing noise and optimizing the volume of collected Syslog data.

Supported Syslog formats

Embedded Collectors support the collection of Syslog-based events from network devices. The collectors support these Syslog message formats:

- RFC 5424
- RFC 3164

The supported Syslog formats are:

- RFC5424 Syslog format
- RFC3164 Syslog format

Syslog collection job outputs

A Syslog collection job output is a record generated by the Syslog collector that:

- reflects the format configuration for the onboarded device
- contains Syslog events received from the device in the specified format, and

- is used to monitor and audit Syslog messages in Crosswork Network Controller.

When onboarding a network device, you must select the appropriate Syslog Format. The chosen format determines whether Syslog events are parsed and displayed as UNKNOWN, RFC5424, or RFC3164.

Syslog collection job output formats (Reference)

1. UNKNOWN: Syslog Collection Job output contains Syslog events as received from device.

Note If the device is configured to generate Syslog events in RFC5424/RFC3164 format but no format is specified in the **Syslog Format** field, this is considered as **UNKNOWN** by default.

Sample output:

```
node_id_str: "xrv9k-VM8"
node_id_uuid: ":i\300\216>\366BM\262\270@\337\225\2723&"
collection_id: 1056
collection_start_time: 1616711596200
msg_timestamp: 1616711596201
data_gpbkv {
  timestamp: 1616711596201
  name: "syslogsensor.path"
  fields {
    name: "RAW"
    string_value: "<6>1 Mar 25 15:34:41.321 PDT - SSHD_ 69570 - - 98949:
RP/0/RP0/CPU0:SSHD_[69570]: %SECURITY-SSHD-6-INFO_SUCCESS : Successfully authenticated
user \'admin\' from \'40.40.40.116\' on \'vty0\'(cipher \'aes128-ctr\', mac \'hmac-sha1\')
 \n"
  }
  fields {
    name: "DEVICE_IP"
    string_value: "40.40.40.30"
  }
}
collection_end_time: 1616711596200
collector_uuid: "17328736-b726-4fe3-b922-231a4a30a54f:SYSLOG_COLLECTOR"
status {
  status: SUCCESS
}
model_data {
}
sensor_data {
  syslog_sensor {
    pris {
      facilities: 0
      facilities: 3
      facilities: 4
      facilities: 23
      severities: 0
      severities: 5
      severities: 6
      severities: 7
    }
  }
}
application_contexts {
  application_id: "SyslogApp-xr-8-job1"
  context_id: "xr-8-job1"
```

```
}
version: "1"
```

2. RFC5424: If the device is configured to generate Syslog events in RFC5424 format and the RFC5424 format is selected in the **Syslog Format** field, the Syslog Job Collection output contains Syslog events as received from device (RAW) and the RFC5424 best-effort parsed Syslog events from the device.

Note The Syslog collector will parse the Syslog event on best efforts as per the following Java RegEx pattern:

RFC5424

```
"^<(?<pri>\\d+)>(?<version>\\d{1,3})\\s*(?<date>(([0-9]{4}\\s+)?[a-zA-Z]{3}\\s+\\d+\\s+\\d+:\\d+:\\d+.\\d{3}\\s+[a-zA-Z]{3}?[:]
9T:.Z-]+))\\s*(?<host>\\S+)\\s*(?<processname>\\S+)\\s*(?<procid>\\S+)\\s*(?<msgid>\\S+)\\s*(?<structureddata>(-|\\[.+\\]))\\
<message>.+)$";
```

Sample output:

```
....
....


collection_start_time: 1596307542398
msg_timestamp: 1596307542405
data_gpbkv {
  timestamp: 1596307542405
  name: "syslogsensor.path"
  fields {
    name: "RAW"
    string_value: "<13>1 2020 Aug  1 12:03:32.461 UTC:  iosxr254node config 65910 - -
2782: RP/0/RSP0/CPU0:2020 Aug  1 12:03:32.461 UTC: config[65910]: %MGBL-SYS-5-CONFIG_I
: Configured from console by admin on vty0 (10.24.88.215) \n"
  }
  fields {
    name: "RFC5424"
    string_value: "pri=13,  severity=5,  facility=1,  version=1,
date=2020-08-01T12:03:32.461,  remoteAddress=/172.28.122.254,  host=\'iosxr254node\',
message=\'2782: RP/0/RSP0/CPU0:2020 Aug  1 12:03:32.461 UTC: config[65910]:
%MGBL-SYS-5-CONFIG_I : Configured from console by admin on vty0 (10.24.88.215) \',
messageId=null, processName=config, structuredDataList=null"
  }
  fields {
    name: "DEVICE_IP"
    string_value: "172.28.122.254"
  }
}
collection_end_time: 1596307542404
collector_uuid: "ac961b09-8f67-4c93-a99a-31eef50f7fa9:SYSLOG_COLLECTOR"
status {
  status: SUCCESS
}
...
...
```

3. RFC3164: If the device is configured to generate Syslog events in RFC3164 format and the RFC3164 format is selected in **Syslog Format** field, the Syslog Job Collection output contains both RAW (as received from device) Syslog events and the RFC3164 best-effort parsed Syslog events from the device.

Note The Syslog collector will parse the Syslog event on best efforts as per the following Java RegEx pattern:

RFC3164

```
"^(<(?<pri>\\d+)>[:]*\\s*)?(?<date>(\\*[a-zA-Z]{3}\\s+\\d+\\s+[0-9]{4}\\s+\\d+:\\d+:\\d+\\.[\\d{3}\\s+]+[[a-zA-Z]{3}[:]*]?\\s+)|(([0-9]{4}
[a-zA-Z]{3}\\s+\\d+\\s+\\d+:\\d+:\\d+[.]*[\\d{3}\\s+]+[[a-zA-Z]{3}[:]*]*)|([0-9T:.Z-]+))\\s+(?<host>\\S+)?\\s+((?<tag>[^\\[\\s\\]]+)(\\[(?
<procid>\\d+)\\])?:)*\\s*(?<message>.+)$";
```

Sample output:

```
....
.....
collection_id: 20
collection_start_time: 1596306752737
msg_timestamp: 1596306752743
data_gpbkv {
  timestamp: 1596306752743
  name: "syslogsensor.path"
  fields {
    name: "RAW"
    string_value: "<14>2020 Aug  1 11:50:22.799 UTC:  iosxr254node 2756:
RP/0/RSP0/CPU0:2020 Aug  1 11:50:22.799 UTC: config[65910]: %MGBL-CONFIG-6-DB_COMMIT :
Configuration committed by user \'admin\'. Use \'show configuration commit changes
1000000580\' to view the changes. \n"
  }
  fields {
    name: "RFC3164"
    string_value: "pri=14,  severity=6,  facility=1,  version=null,
date=2020-08-01T11:50:22.799,  remoteAddress=/172.28.122.254,  host=\'iosxr254node\',
message=\'RP/0/RSP0/CPU0:2020 Aug  1 11:50:22.799 UTC: config[65910]:
%MGBL-CONFIG-6-DB_COMMIT : Configuration committed by user \'admin\'. Use \'show
configuration commit changes 1000000580\' to view the changes. \', tag=2756"
  }
  fields {
    name: "DEVICE_IP"
    string_value: "172.28.122.254"
  }
}
collection_end_time: 1596306752742
collector_uuid: "ac961b09-8f67-4c93-a99a-31eef50f7fa9:SYSLOG_COLLECTOR"
status {
  status: SUCCESS
}
....
....
```

Warning If the Syslog collector is unable to parse the Syslog events according to the format specified in the **Syslog Format** field, then the Syslog Collection Job output contains Syslog events as received from device (RAW).

Device configuration for non-secure Syslog

This section presents configurations for non-secure Syslog operation in Cisco IOS XR and IOS XE platforms, supporting both RFC3164 and RFC5424 message formats. Use the configurations below to achieve correct syslog parsing and message delivery.

Configure RFC3164 syslog format

For IOS XR:

```
logging <Data IP> port 30514 OR logging <Data IP> vrf <vrfname> port 30514
logging trap [severity]
logging facility [facility value]
logging suppress duplicates
service timestamps log datetime msec show-timezone year
logging hostnameprefix <some host related prefix e.g.iosxrhost2>
```

Note Ensure “service timestamps log…” and “logging hostnameprefix…” are present.

For IOS XE:

```
no logging message-counter syslog
logging trap <severity>
logging facility <facility>
logging host <Data IP> transport tcp port 309898 session-id string <sessionidstring> -->
To use TCP channel
OR
logging host <Data IP> transport udp port 30514 session-id string <sessionidstring> --->
To use UDP channel
OR
logging host <Data IP> vrf Mgmt-intf transport udp port 30514 session-id string
<sessionidstring> --> To use UDP via vrf
service timestamps log datetime msec year show-timezone
```

Note TCP transports require explicit configuration as shown.

Configure RFC5424 syslog format

For IOS XR:

```
logging <Data IP> port 30514 OR logging <server 1> vrf <vrfname> port 30514
logging trap [severity]
logging facility [facility value]
logging suppress duplicates
service timestamps log datetime msec show-timezone year
logging hostnameprefix <some host related prefix e.g.iosxrhost2>
logging format rfc5424
```

Note Ensure “service timestamps log…” and “logging hostnameprefix…” are present.

For IOS XE:

```
no logging message-counter syslog
logging trap <severity>
logging facility <facility>
logging host <Data IP> transport tcp port 309898 session-id string <sessionidstring> -->
To use TCP channel
OR
logging host <Data IP> transport udp port 30514 session-id string <sessionidstring> --->
To use UDP channel
OR
```

```
logging host <Data IP> vrf Mgmt-intf transport udp port 30514 session-id string
<sessionidstring> --> To use UDP via vrf
service timestamps log datetime msec year show-timezone
logging trap syslog-format 5424 --> if applicable
```

Note TCP transports require explicit configuration as shown.

Device configuration for secure Syslog

This section presents configurations for secure Syslog operation in Cisco IOS XR and IOS XE platforms, supporting both RFC3164 and RFC5424 message formats. Use the configurations below to achieve correct syslog parsing and message delivery.

Use the steps to establish a secured syslog communication with the device.

- Download the Cisco Crosswork trust chain from the **Certificate Management UI** page in Cisco Crosswork. See Download syslog certificates, on page 44.
- Configure the device with the Cisco Crosswork trustchain. See Configure a Crosswork trustpoint on a device, on page 44.

Download syslog certificates

Provide users with the ability to download syslog certificates from the Crosswork Network Controller UI.

This task helps administrators securely export device syslog certificates for system integration, troubleshooting, or compliance requirements.

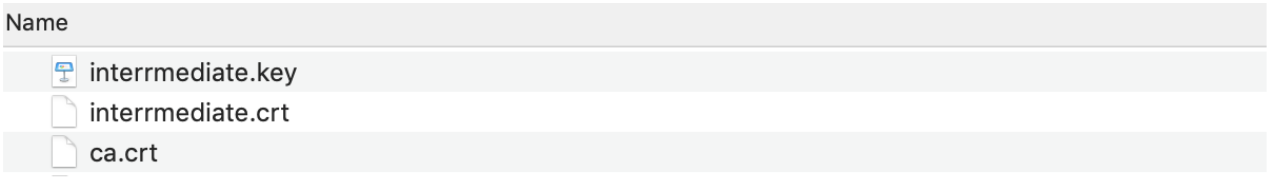
Procedure

Step 1 In the Crosswork Network Controller UI, go to **Administration > Certificate Management**.

Step 2 Click the *i* icon in the Crosswork-Device-Syslog row.

Step 3 Click **Export All** to download the certificates.

The files are downloaded to your system.



The files are downloaded to your system.

Configure a Crosswork trustpoint on a device

This procedure enables secure, trusted communication (TLS/PKI) between a Cisco device (IOS XR or IOS XE) and Crosswork applications using trustpoints.

When integrating network devices with Cisco Crosswork monitoring for authenticated, encrypted syslog export.

Procedure

Step 1 Enable TLS by configuring the IOS XR or XE device (refer to these samples):

- For IOS XR:

```
RP/0/RSP0/CPU0:ASR9k(config)#crypto ca trustpoint syslog-root
RP/0/RSP0/CPU0:ASR9k(config-trustp)#enrollment terminal
RP/0/RSP0/CPU0:ASR9k(config-trustp)#crl optional
RP/0/RSP0/CPU0:ASR9k(config-trustp)#commit
RP/0/RSP0/CPU0:ASR9k(config-trustp)#end
RP/0/RSP0/CPU0:ASR9k#
RP/0/RSP0/CPU0:ASR9k#crypto ca authenticate syslog-root
Fri Jan 22 11:07:41.880 GMT


Enter the base 64 encoded certificate.
End with a blank line or the word "quit" on a line by itself

-----BEGIN CERTIFICATE-----
MIIGKzCCBBOgAwIBAgIRAKfyU89yjmrXVDRKBWuSGPgwDQYJKoZIhvcNAQELBQAw
bDELMAkGA1UEBhMCVVMxCzAJBgNVBAgTAkNBMREwDwYDVQQHEwhTYW4gSm9zZTEa
.............................................................
.............................................................
jPQ/UrO8N3sC1gGJX7CIIh5cE+KIJ51ep8i1eKSJ5wHWRTmv342MnG2StgOTtaFF
vrkWHD02o6jRuYXDWEUptDOg8oEritZb+SNPXWUc/2mbYog6ks6EeMC69VjkZPo=
-----END CERTIFICATE-----

Read 1583 bytes as CA certificate
  Serial Number  : A7:F2:53:CF:72:8E:6A:D7:54:34:4A:05:6B:92:18:F8
  Subject:
               CN=Crosswork Device Root CA,O=CISCO SYSTEMS INC,L=San Jose,ST=CA,C=US
  Issued By      :
               CN=Crosswork Device Root CA,O=CISCO SYSTEMS INC,L=San Jose,ST=CA,C=US
  Validity Start : 02:37:09 UTC Sat Jan 16 2021
  Validity End   : 02:37:09 UTC Thu Jan 15 2026
  SHA1 Fingerprint:
               209B3815271C22ADF78CB906F6A32DD9D97BBDBA

Fingerprint: 2FF85849EBAAB9B059ACB9F5363D5C9CDo you accept this certificate? [yes/no]: yes
RP/0/RSP0/CPU0:ASR9k#config
RP/0/RSP0/CPU0:ASR9k(config)#crypto ca trustpoint syslog-inter
RP/0/RSP0/CPU0:ASR9k(config-trustp)#enrollment terminal
RP/0/RSP0/CPU0:ASR9k(config-trustp)#crl optional
RP/0/RSP0/CPU0:ASR9k(config-trustp)#commit
RP/0/RSP0/CPU0:ASR9k#crypto ca authenticate syslog-inter
Fri Jan 22 11:10:30.090 GMT


Enter the base 64 encoded certificate.
End with a blank line or the word "quit" on a line by itself

-----BEGIN CERTIFICATE-----
MIIGFDCCA/ygAwIBAgIRAKhqHQXcJzQzeQK6U2wn8PIwDQYJKoZIhvcNAQELBQAw
bDELMAkGA1UEBhMCVVMxCzAJBgNVBAgTAkNBMREwDwYDVQQHEwhTYW4gSm9zZTEa
.............................................................
.............................................................
5lBk617z6cxFER5c+/PmJFhcreisTxXg1aJbFdnB5C8f+0uUIdLghykQ/zaZGuBn
```

```
AAB70c9r9OeKGJWzvv1e2U8HH1pdQ/nd
-----END CERTIFICATE-----

Read 1560 bytes as CA certificate
  Serial Number  : 02:48:6A:1D:05:DC:27:34:33:79:02:BA:53:6C:27:F0:F2
  Subject:
               CN=device-syslog,O=CISCO SYSTEMS INC,L=San Jose,ST=CA,C=US
  Issued By      :
               CN=Crosswork Device Root CA,O=CISCO SYSTEMS INC,L=San Jose,ST=CA,C=US
  Validity Start : 02:37:11 UTC Sat Jan 16 2021
  Validity End   : 02:37:11 UTC Mon Jan 16 2023
  SHA1 Fingerprint:
               B06F2BFDE95413A8D08A01EE3511BC3D42F01E59

CA Certificate validated using issuer certificate.
RP/0/RSP0/CPU0:ASR9k#show crypto ca certificates
Fri Jan 22 15:45:17.196 GMT


Trustpoint       : syslog-root
================================================
CA certificate
  Serial Number  : A7:F2:53:CF:72:8E:6A:D7:54:34:4A:05:6B:92:18:F8
  Subject:
        CN=Crosswork Device Root CA,O=CISCO SYSTEMS INC,L=San Jose,ST=CA,C=US
  Issued By      :
        CN=Crosswork Device Root CA,O=CISCO SYSTEMS INC,L=San Jose,ST=CA,C=US
  Validity Start : 02:37:09 UTC Sat Jan 16 2021
  Validity End   : 02:37:09 UTC Thu Jan 15 2026
  SHA1 Fingerprint:
         209B3815271C22ADF78CB906F6A32DD9D97BBDBA


Trustpoint       : syslog-inter
================================================
CA certificate
  Serial Number  : 02:48:6A:1D:05:DC:27:34:33:79:02:BA:53:6C:27:F0:F2
  Subject:
        CN=device-syslog,O=CISCO SYSTEMS INC,L=San Jose,ST=CA,C=US
  Issued By      :
        CN=Crosswork Device Root CA,O=CISCO SYSTEMS INC,L=San Jose,ST=CA,C=US
  Validity Start : 02:37:11 UTC Sat Jan 16 2021
  Validity End   : 02:37:11 UTC Mon Jan 16 2023
  SHA1 Fingerprint:
         B06F2BFDE95413A8D08A01EE3511BC3D42F01E59
RP/0/RSP0/CPU0:ASR9k(config)#logging tls-server syslog-tb131
RP/0/RSP0/CPU0:ASR9k(config-logging-tls-peer)#tls-hostname 10.13.0.159
RP/0/RSP0/CPU0:ASR9k(config-logging-tls-peer)#trustpoint syslog-inter
RP/0/RSP0/CPU0:ASR9k(config-logging-tls-peer)#severity debugging
RP/0/RSP0/CPU0:ASR9k(config-logging-tls-peer)#vrf default
RP/0/RSP0/CPU0:ASR9k(config-logging-tls-peer)#commit
RP/0/RSP0/CPU0:ASR9k(config-logging-tls-peer)#exit
RP/0/RSP0/CPU0:ASR9k(config)#exit
RP/0/RSP0/CPU0:ASR9k#exit
RP/0/RSP0/CPU0:ASR9k#show running-config logging
Fri Jan 22 11:17:19.385 GMT
logging tls-server syslog-tb131
vrf default
severity debugging
trustpoint syslog-inter
tls-hostname <Device Southbound IP>
!
logging trap debugging
logging format rfc5424
```

```
logging facility user
logging hostnameprefix ASR9k
logging suppress duplicates

RP/0/RSP0/CPU0:ASR9k#
```

- For IOS XE:

```
csr8kv(config)#crypto pki trustpoint syslog-root
csr8kv(ca-trustpoint)#enrollment terminal
csr8kv(ca-trustpoint)#revocation-check none
csr8kv(ca-trustpoint)#chain-validation stop
csr8kv(ca-trustpoint)#end
csr8kv(config)#crypto pki authenticate syslog-root

Enter the base 64 encoded CA certificate.
End with a blank line or the word "quit" on a line by itself

-----BEGIN CERTIFICATE-----
MIIFPjCCAyYCCQCO6pK5AOGYdjANBgkqhkiG9w0BAQsFADBhMQswCQYDVQQGEwJV
UzELMAkGA1UECAwCQ0ExETAPBgNVBAcMCE1pbHBpdGFzMQ4wDAYDVQQKDAVDaXNj
.....................................................................
.....................................................................
JbimOpXAncoBLo14DXOJLvMVRjn1EULE9AXXCNfnrnBx7jL4CV+qHgEtF6oqclFW
JEA=
-----END CERTIFICATE-----

Certificate has the following attributes:
       Fingerprint MD5: D88D6D8F E53750D4 B36EB498 0A435DA1
      Fingerprint SHA1: 649DE822 1C222C1F 5101BEB8 B29CDF12 5CEE463B

% Do you accept this certificate? [yes/no]: yes
Trustpoint CA certificate accepted.
% Certificate successfully imported


csr8kv(config)#crypto pki trustpoint syslog-intermediate
csr8kv(ca-trustpoint)#enrollment terminal
csr8kv(ca-trustpoint)#revocation-check none
csr8kv(ca-trustpoint)#chain-validation continue syslog-root
csr8kv(ca-trustpoint)#end
csr8kv(config)#crypto pki authenticate syslog-intermediate

Enter the base 64 encoded CA certificate.
End with a blank line or the word "quit" on a line by itself

-----BEGIN CERTIFICATE-----
MIIFfTCCA2WgAwIBAgICEAAwDQYJKoZIhvcNAQELBQAwXDELMAkGA1UEBhMCVVMx
EzARBgNVBAgMCkNhbGlmb3JuaWExDjAMBgNVBAoMBUNpc2NvMQ4wDAYDVQQLDAVT
.....................................................................
.....................................................................
Nmz6NQynD7bxdQa9Xq9kyPuY3ZVKXkf312IRH0MEy2yFX/tAen9JqOeZ1g8canmw
TxsWA5TLzy1RmxqQh88f0CM=
-----END CERTIFICATE-----
Trustpoint 'syslog-intermediate' is a subordinate CA.
but certificate is not a CA certificate.
Manual verification required
Certificate has the following attributes:
       Fingerprint MD5: FE27BDBE 9265208A 681670AC F59A2BF1
      Fingerprint SHA1: 03F513BD 4BEB689F A4F4E001 57EC210E 88C7BD19

csr8kv(config)#logging host <Device Southbound IP> transport tls port 30614
csr8kv(config)#logging trap informational syslog-format rfc5424
csr8kv(config)#logging facility user
csr8kv(config)#service timestamps log datetime msec year show-timezone
```

```
csr8kv(config)#logging tls-profile tlsv12
```

Step 2 If configuring for FQDN, perform these additional steps:

a) Configure the domain name and DNS IP on the device:

- For IOS XR:

```
RP/0/RSP0/CPU0:ASR9k#config
RP/0/RSP0/CPU0:ASR9k(config)#domain name <DNS domain name>
RP/0/RSP0/CPU0:ASR9k(config)#domain name-server <DNS server IP>
```

- For IOS XE:

```
Device(config)# ip name-server <IP of DNS>
Device(config)# ip domain name <domain name>
```

b) Configure Embedded Collectors VIP FQDN for `tls-hostname`:

- For IOS XR:

```
RP/0/RSP0/CPU0:ASR9k(config)#logging tls-server syslog-tb131
RP/0/RSP0/CPU0:ASR9k(config-logging-tls-peer)#tls-hostname <Device VIP FQDN>
```

- For IOS XE:

```
Device(config)# logging host fqdn ipv4 <hostname> transport tls port 30614
```

The device is now securely configured with a Crosswork trustpoint, enabling authenticated and encrypted log exports using TLS.

gNMI collection jobs

A gNMI telemetry collection is a data collection mechanism that:

- uses the gRPC Network Management Interface (gNMI) protocol via Embedded Collectors to gather telemetry data from devices
- supports only the gNMI Dial-In (gRPC Dial-In) streaming telemetry model based on subscriptions, relays subscription responses (notifications) to designated destinations, and
- relies on model compatibility with the target device platform. Devices require gNMI configuration before collection jobs can be initiated.

In the gNMI operation, Crosswork Network Controller can use both secure and insecure connection modes, with preference dictated by inventory settings. If a device reloads, the gNMI collector automatically resubscribes to maintain ongoing data collection. The gNMI specification does not define message termination, so some cadence controls are unsupported for gNMI collectors.

The gNMI specification does not define a message termination mechanism; therefore, Destination and Dispatch cadence settings are unsupported for gNMI collectors. The cadence parameter that controls polling frequency for Embedded Collectors is not applicable in gNMI-based collection.

Additional reference information

Supported gNMI subscribtion options

Embedded Collectors support multiple subscription paths per device, allowing combinations of ON_CHANGE and ONCE collection jobs:

- ON_CHANGE: Collects and delivers data only when specified elements change.
- ONCE: Collects and sends a one-time snapshot of the current data for the specified path.
- SAMPLE (under STREAM): Collects data at specified intervals if the device supports cadence-based collection.
- TARGET_DEFINED: The device determines the mode for each path (SAMPLE or ON_CHANGE) according to its capabilities.

Table 7: gNMI subscription options

Type	Subtype	Description
Once		Collects and sends the current snapshot of the system configuration only once for all specified paths.
Stream	SAMPLE	Cadence-based collection.
	ON_CHANGE	Sends initial state, then updates when the subscribed data changes.
	TARGET_DEFINED	Router or device selects mode (SAMPLE or ON_CHANGE) per path according to device configuration. Router/Device chooses the mode of subscription on a per-leaf basis based on the subscribed path (i.e. one of SAMPLE or ON_CHANGE)

Note

- Embedded Collectors depend on the device to declare supported subscription modes.
- gNMI sensor paths with default values do not appear in the payload due to Protocol Buffers conventions (e.g., default bool is false).

  For boolean the default value is false. For enum, it is gnmi.proto specified.

  Example:

```
message GNMIDeviceSetting {
bool suppress_redundant = 1;
bool allow_aggregation = 4;
bool updates_only = 6;
}
```

  Example:

```
enum SubscriptionMode {
TARGET_DEFINED = 0; //default value will not be printed
ON_CHANGE = 1;
SAMPLE = 2;
}
```

Sample gNMI collection payload

In this sample you see two collections for the device group "milpitas". The first job collects interface statistics, every 60 seconds using the "mode" = "SAMPLE". The second job captures any changes to the interface state (up/down). If this is detected, it is simply sent "mode" = "STREAM" to the collector.

```
{
    "collection_job": {
        "job_device_set": {
            "device_set": {
                "device_group": "milpitas"
            }
        },
        "sensor_output_configs": [{
            "sensor_data": {
                "gnmi_standard_sensor": {
                    "Subscribe_request": {
                        "subscribe": {
                            "subscription": [{
                                "path": {
                                    "origin": "openconfig-interfaces",
                                    "elem": [{
                                        "name": "interfaces/interface/state/ifindex"
                                    }]
                                },
                                "mode": "SAMPLE",
                                "sample_interval": 10000000000
                            }, {
                                "path": {
                                    "origin": "openconfig-interfaces",
                                    "elem": [{
                                        "name":
"interfaces/interfaces/state/counters/out-octets"
                                    }]
                                },
                                "mode": "ON_CHANGE",
                                "sample_interval": 10000000000
```

```
                                }],
                                "mode": "STREAM",
                                "encoding": "JSON"
                            }
                        }
                    },
                    "destination": {
                        "context_id": "hukarz",
                        "destination_id": "c2a8fba8-8363-3d22-b0c2-a9e449693fae"
                    }
                }],
                "sensor_input_configs": [{
                    "sensor_data": {
                        "gnmi_standard_sensor": {
                            "Subscribe_request": {
                                "subscribe": {
                                    "subscription": [{
                                        "path": {
                                            "origin": "openconfig-interfaces",
                                            "elem": [{
                                                "name": "interfaces/interface/state/ifindex"
                                            }]
                                        },
                                        "mode": "SAMPLE",
                                        "sample_interval": 10000000000
                                    }, {
                                        "path": {
                                            "origin": "openconfig-interfaces",
                                            "elem": [{
                                                "name":
"interfaces/interfaces/state/counters/out-octets"
                                            }]
                                        },
                                        "mode": "ON_CHANGE",
                                        "sample_interval": 10000000000
                                    }],
                                    "mode": "STREAM",
                                    "encoding": "JSON"
                                }
                            }
                        }
                    },
                    "cadence_in_millisec": "60000"
                }],
                "application_context": {
                    "context_id": "testing.group.gnmi.subscription.onchange",
                    "application_id": "testing.postman.gnmi.standard.persistent"
                },
                "collection_mode": {
                    "lifetime_type": "APPLICATION_MANAGED",
                    "collector_type": "GNMI_COLLECTOR"
                }
            }
        }
```

Enable secure gNMI communication between a device and Crosswork

Enable secure gNMI data exchanges between a device and Cisco Crosswork using certificate-based authentication.

Cisco Crosswork accepts a single rootCA for signing device certificates. All device certificates must be signed by the same CA, ensuring trusted and secure communication.

Before you begin

Ensure you have required certificate files (root CA, device certificates, device key).

Use these steps to enable secure gNMI between Cisco Crosswork and the devices.

Procedure

Step 1 Generate certificates for the device, signed by the rootCA trusted by Crosswork. See Generate the device certificates, on page 52.

Step 2 Upload the certificates to the Crosswork Certificate Management UI. See Configure the gNMI certificate, on page 54.

Step 3 Update device configuration with secure gNMI port details provided by Crosswork. See Update protocol on device from Crosswork.

Step 4 Enable gNMI on the device. See Configure devices for gNMI-based telemetry, on page 55.

Step 5 Enable gNMI bundling on the device. See Configure gNMI bundling for IOS XR, on page 58.

Step 6 Configure the certificates and device key on the device. See Certificate management for IOS XR and XE devices, on page 59.

The device and Cisco Crosswork establish a secure, certificate-authenticated gNMI connection.

Generate the device certificates

Generate device certificates using OpenSSL for secure communication between devices and certificate authorities.

The certificate generation procedure has been validated using both OpenSSL and Microsoft tools. Contact Cisco Support if using a different tool.

Note To generate device certificates using a different utility, contact the Cisco Support team.

Before you begin

- Install OpenSSL on your system.
- Ensure you have access permissions for certificate storage.

Procedure

Step 1 Create the rootCA certificate.

a) Set the desired validity duration of the certificates using the “days” parameter (recommended: 365 or more).

```
# openssl genrsa -out rootCA.key
# openssl req -subj /C=/ST=/L=/O=/CN=CrossworkCA -x509 -new -nodes -key rootCA.key -sha256 -out
rootCA.pem -days 1024
```

Step 2 Create device key and certificate.

```
# openssl genrsa -out device.key
# openssl req -subj /C=/ST=/L=/O=/CN=Crosswork -new -key device.key -out device.csr
# openssl x509 -req -extfile <(printf "subjectAltName=IP.0: 10.58.56.18") -in device.csr -CA rootCA.pem
 -CAkey rootCA.key -CAcreateserial -sha256 -out device.crt -days 1024
```

a) For multiple devices, specify multiple IP addresses in the subjectAltName, separated by commas in `subjectAltName`.

```
# openssl x509 -req -extfile <(printf "subjectAltName=IP.0: 10.58.56.18, IP.1: 10.58.56.19, IP.2:
 10.58.56.20 ..... ") -in device.csr -CA rootCA.pem -CAkey rootCA.key -CAcreateserial -sha256 -out
 device.crt -days 1024
```

Step 3 Verify the certificate contains the expected Subject Alternative Name (SAN) details.

```
# openssl x509 -in device.crt -text -noout
```

The system generates a device certificate with the designated subject alternative names, enabling secure communication.

Sample output:

```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            66:38:0c:59:36:59:da:8c:5f:82:3b:b8:a7:47:8f:b6:17:1f:6a:0f
        Signature Algorithm: sha256WithRSAEncryption
        Issuer: CN = rootCA
        Validity
            Not Before: Oct 28 17:44:28 2021 GMT
            Not After : Aug 17 17:44:28 2024 GMT
        Subject: CN = Crosswork
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                RSA Public-Key: (2048 bit)
                Modulus:
                    00:c6:25:8a:e8:37:7f:8d:1a:7f:fa:e2:d6:10:0d:
                    b8:e6:2b:b0:b0:7e:ab:c9:f9:14:a3:4f:2e:e6:30:
                    97:f4:cd:d6:11:7d:c0:a6:9b:43:83:3e:26:0f:73:
                    42:89:3c:d7:62:7b:04:af:0b:16:67:4c:8e:60:05:
                    cc:dd:99:37:3f:a4:17:ed:ff:28:21:20:50:6f:d9:
                    be:23:78:07:dc:1e:31:5e:5f:ca:54:27:e0:64:80:
                    03:33:f1:cd:09:52:07:6f:13:81:1b:e1:77:e2:08:
                    9f:b4:c5:97:a3:71:e8:c4:c8:60:18:fc:f3:be:5f:
                    d5:37:c6:05:6e:9e:1f:65:5b:67:46:a6:d3:94:1f:
                    38:36:54:be:23:28:cc:7b:a1:86:ae:bd:0d:19:1e:
                    77:b7:bd:db:5a:43:1f:8b:06:4e:cd:89:88:e6:45:
                    0e:e3:17:b3:0d:ba:c8:25:9f:fc:40:08:87:32:26:
                    69:62:c9:57:72:8a:c2:a1:37:3f:9d:37:e9:69:33:
                    a5:68:0f:8f:f4:31:a8:bc:34:93:a3:81:b9:38:87:
                    2a:87:a3:4c:e0:d6:aa:ad:a7:5c:fb:98:a2:71:15:
                    68:e7:8d:0f:71:9a:a1:ca:10:81:f8:f6:85:86:c1:
                    06:cc:a2:47:16:89:ee:d1:90:c9:51:e1:0d:a3:2f:
                    9f:0b
                Exponent: 65537 (0x10001)
        X509v3 extensions:
            X509v3 Subject Alternative Name:
                IP Address:10.58.56.18
    Signature Algorithm: sha256WithRSAEncryption
         01:41:2c:91:0b:a1:10:8a:11:1a:95:36:99:2c:27:31:d3:7d:
         e9:4b:29:56:c3:b7:00:8c:f4:39:d2:8c:50:a4:da:d4:96:93:
         eb:bb:71:e3:70:d3:fe:1f:97:b2:bc:5c:f8:f4:65:ed:83:f7:
         67:56:db:0f:67:c2:3d:0c:e7:f8:37:65:1d:11:09:9a:e3:42:
```

```
bc:c6:a0:31:7c:1f:d7:5e:c6:86:72:43:a8:c1:0c:70:33:60:
dc:14:5b:9d:f3:ab:3d:d5:d2:94:90:1c:ba:fd:80:4d:22:e3:
31:93:c7:16:5f:85:20:38:ad:36:b9:1a:e0:89:8e:06:8c:f8:
cd:55:cc:a1:89:d3:91:7f:66:61:a3:40:71:c2:1e:ee:3b:80:
37:af:73:5e:8e:0d:db:4b:49:da:a6:bd:7d:0a:aa:9e:9a:9e:
fa:ed:05:25:08:f2:4d:cd:2f:63:55:cf:be:b1:5d:03:c2:b3:
32:bf:f4:7b:1a:10:b9:5e:69:ac:77:5e:4a:4f:85:e3:7f:fe:
04:df:ce:3e:bb:28:8f:e3:bf:1a:f9:0f:94:18:08:86:7d:59:
57:71:0a:97:0d:86:9c:63:e7:0e:48:7d:f0:0e:1d:67:ff:9b:
1d:1b:05:25:c8:c3:1f:f4:52:0f:e1:bf:86:d7:ec:47:10:bd:
94:cf:ca:e2
```

Configure the gNMI certificate

Upload and configure a gNMI certificate in Cisco Crosswork to enable secure device communication via gNMI.

gNMI collectors operate as clients and access devices (gNMI servers). They require a valid certificate for trust validation. The certificate establishes a trusted chain for device authentication.

Before you begin

- Ensure you have the root CA (.pem) or relevant trust chain file available.
- Gather the necessary certificate details (such as name and role).
- Verify that the trust chain in your .pem file includes all devices participating in gNMI communication.
- Be aware that you can upload only one gNMI certificate to Crosswork.

Procedure

Step 1 In the Cisco Crosswork UI, go to **Administration > Certificate Management**.

Step 2 Click the + icon to add the certificate.

Step 3 In the **Add Certificate** window, provide these details:

a) **Device Certificate Name**: Enter a name for the certificate.
b) Certificate Role: Select "Device gNMI/gRPC communication".
c) Device Trust Chain: Upload the Root CA file or trust chain (.pem file).

If you have multiple trust chains, add all of them (across vendors) to a single .pem file and upload that.

Step 4 Click **Save**.

The newly added gNMI certificate appears in the configured certificates list.

Update protocol on device from Crosswork

Perform the update of protocol details for a specified device using the Crosswork Network Controller.

Use this task to establish or update device communication security, particularly after certificate configuration or when maintaining compliance with network security protocols.

Before you begin

- Ensure you have configured the gNMI certificate in Crosswork Network Controller.

- Prepare the CSV file containing device details, including the required protocol information.

Procedure

Step 1 After configuring the gNMI certificate in Crosswork Network Controller, ensure your device is listed in the network inventory.

Step 2 Update the device with secure protocol details:

a) Use the Cisco Crosswork UI at **Device Management > Network Devices**.
b) Specify protocol details as GNMI_SECURE Port in the CSV file corresponding to the device.
c) Import the device entry or edit it if necessary.
d) View the **Edit Device** page for confirmation.

The device is updated with the specified secure protocol settings, and the Crosswork inventory should reflect the new configurations.

What to do next

- Validate device communication and ensure connectivity using updated protocols.
- Monitor alerts and error messages related to protocol changes.

Configure devices for gNMI-based telemetry

This section provides instruction on how to enable both platforms to collect and export telemetry data using gNMI.

Enable gNMI-based telemetry data collection on IOS XR and IOS XE devices.

- On Cisco IOS XR devices, enable the gNMI service and specify telemetry parameters. See Configure IOS XR device for gNMI, on page 55.
- On Cisco IOS XE devices, enable gNMI and configure required access credentials. See Configure IOS XE device for gNMI, on page 57.

Configure IOS XR device for gNMI

Configure IOS XR devices to enable gNMI-based telemetry data collection.

Use these instructions to enable gNMI on IOS XR devices so telemetry collectors can stream data via gRPC over HTTP/2.

Before you begin

- Confirm the network environment supports gRPC over HTTP/2.
- Obtain any necessary port numbers and security certificates.

Procedure

Step 1 Enable gRPC on an HTTP/2 connection.

a) Enter configuration mode and activate gRPC with a specified port:

Example:

```
Router#configure
Router(config)#grpc
Router(config-grpc)#port <port-number>
```

b) Specify a port number within the range 57344 to 57999. If the port number is unavailable, an error will appear.

Step 2 Configure session parameters.

where:

a) Use this command structure:

Example:

```
Router(config)#grpc{ address-family | dscp | max-request-per-user | max-request-total | max-streams
 |
max-streams-per-user | no-tls | service-layer | tls-cipher | tls-mutual | tls-trustpoint | vrf }
```

b) Parameter options:

Table 8: Parameters and descriptions

Parameters	Descriptions
`address-family`	Set the address-family identifier type.
`dscp`	Set the DSCP QoS marking on transmitted gRPC traffic.
`max-request-per-user`	Set the maximum concurrent requests per user.
`max-request-per-user`	Set the maximum concurrent requests per user.
`max-request-total`	Set the maximum concurrent requests in total.
`max-streams`	Set the maximum number of concurrent gRPC requests. The maximum subscription limit is 128 requests. The default is 32 requests.
`max-streams-per-user`	Set the maximum concurrent gRPC requests for each user. The maximum subscription limit is 128 requests. The default is 32 requests.
`no-tls`	Disable transport layer security (TLS). TLS is enabled by default.
`service-layer`	Enable gRPC service layer configuration.
`tls-cipher`	Enable gRPC TLS cipher suites.
`tls-mutual`	Enable mutual authentication.

Parameters	Descriptions
`tls-trustpoint`	Configure trustpoint.
`server-vrf`	Enable server VRF.

Step 3 Enable Traffic Protection for Third-Party Applications (TPA) on the device.

a) Configure the following settings:

Example:

```
tpa
vrf default
  address-family ipv4
   default-route mgmt
   update-source dataports MgmtEth0/RP0/CPU0/0
```

The IOS XR device is configured to support secure, scalable, gNMI-based telemetry streaming as required for modern network data collection.

Configure IOS XE device for gNMI

Configure IOS XE devices to enable gNMI-based telemetry data collection.

Use these instructions to enable gNMI on IOS XE devices so telemetry collectors can stream data via gRPC over HTTP/2.

Before you begin

- Confirm that your network environment supports gRPC over HTTP/2.
- Obtain required port numbers for gNMI telemetry.
- Acquire security certificates if you are enabling secure gNMI.

Procedure

Step 1 Access device CLI in global configuration mode.

Step 2 Enable the gNMI server in the insecure mode.

Example:

```
Device# configure terminal
Device(config)# gnmi-yang
Device(config)# gnmi-yang server
Device(config)# gnmi-yang port 50000 <The default port is 50052.>
Device(config)# end
Device
```

Step 3 Enable the gNMI server in the secure mode.

Example:

```
Device# configure terminal
Device(config)# gnmi-yang server
Device(config)# gnmi-yang secure-server
```

```
Device(config)# gnmi-yang secure-trustpoint trustpoint1
Device(config)# gnmi-yang secure-client-auth
Device(config)# gnmi-yang secure-port 50001 <The default port is 50051.>
Device(config)# end
Device
```

The IOS XE device is now configured to stream telemetry data using gNMI over gRPC/HTTP2. Telemetry collectors can connect using the configured insecure or secure port.

What to do next

Validate the gNMI connection from your telemetry collector.

Configure gNMI bundling for IOS XR

Enable and configure gNMI bundling capability to optimize telemetry updates on IOS XR devices.

gNMI bundling allows IOS XR to combine multiple Update messages into a single Notification message within a SubscribeResponse. This improves efficiency by reducing the number of notifications sent. You must enable bundling and specify the message size on the IOS XR device.

Before you begin

The gNMI bundling capability can only be configured from the device. It is not available in the Crosswork Interface. Details on how the bundling feature works are available in the Programmability Configuration Guide for Cisco 8000 Series Routers, IOS XR Release 7.8.x.

Procedure

Step 1 Enable gNMI bundling:

```
telemetry model-driven
 gnmi
  bundling
```

Note
The gNMI bundling capability is disabled by default.

Step 2 Set the bundling size, as needed:

```
telemetry model-driven gnmi bundling size<1024-65536>
```

The default bundling size is 32768 bytes.

Note
After processing the (N - 1)th instance, if the message size is less than the configured bundling size, one more instance may be included. This may result in the bundling size being exceeded.

The IOS XR device transmits bundled Update messages according to the specified message size, optimizing telemetry update delivery.

What to do next

To verify bundling is enabled and configured, use:

```
RP/0/RP0/CPU0:R0(config)#telemetry model-driven
RP/0/RP0/CPU0:R0(config-model-driven)#gnmi ?
  bundling   gNMI bundling of telemetry updates
  heartbeat  gNMI heartbeat
  <cr>
RP/0/RP0/CPU0:R0(config-model-driven)#gnmi bundling ?
  size  gNMI bundling size (default: 32768)
  <cr>
RP/0/RP0/CPU0:R0(config-model-driven)#gnmi bundling
RP/0/RP0/CPU0:R0(config-gnmi-bdl)#size ?
  <1024-65536>  gNMI bundling size (bytes)
```

Certificate management for IOS XR and XE devices

This section describes how to import and install certificates on the IOS XR and XE devices. Certificates and trustpoint are only required for secure gNMI servers.

Certificate management encompasses the procedures and requirements for securing communications between Cisco IOS XR and IOS XE devices:

- Cisco IOS XR: Requires certificate and trustpoint configuration for enabling secure gNMI. See Install certificates on a Cisco IOS XR device, on page 59.
- Cisco IOS XE: Similarly requires certificate and trustpoint setup for secure gNMI operations. See Install certificates on a Cisco IOS XE device, on page 60.

Install certificates on a Cisco IOS XR device

Import and install device certificates to enable secure gNMI server operation on Cisco IOS XR.

Certificates and a trustpoint are required only when configuring secure gNMI servers. This procedure guides you through all necessary installation steps.

Before you begin

- Ensure you have the following files ready: `rootCA.pem`, `device.key`, and `device.crt`.
- Ensure access to the IOS XR device filesystem.

Use these steps to install certificates on a Cisco IOS XR device.

Procedure

Step 1 Copy the rootCA.pem, device.key, and device.crt to the device under the `/tmp` folder.

Step 2 Log in to the IOS XR device.

Step 3 Enter the VM shell mode.

Example:

```
RP/0/RP0/CPU0:xrvr-7.2.1#run
```

Step 4 Navigate to the `/grpc` directory.

Example:

```
cd /misc/config/grpc
```

Step 5 Create or replace the contents of these files in `/misc/config/grpc`.

Note
If TLS was previously enabled on your device, these files will already be present in which case replace the content of these files as explained below. If this is the first time, you are enabling TLS on the device, copy the files from the `/tmp` folder to this folder.

a) If enabling TLS for the first time, copy the following from `/tmp`:

- ems.pem with device.crt
- ems.key with device.key
- ca.cert with rootCA.pem

b) If TLS was previously enabled, overwrite these files with the new content.

Step 6 Restart TLS on the device to activate changes:

a) Disable TLS with the `no-tls` command.
b) Re-enable TLS with the `no no-tls` configuration command.

Device certificates are installed, and TLS is restarted. The IOS XR device is ready for secure gNMI operations.

Install certificates on a Cisco IOS XE device

Enable secure gNMI server operation on a Cisco IOS XE device by installing the required certificates and configuring a trustpoint.

Certificates and a trustpoint are required only when configuring secure gNMI servers. This procedure guides you through all necessary installation steps.

Before you begin

- Plan and note the trustpoint name and password you will use.

  Obtain these files:

  - The root CA certificate (e.g., rootCA.pem)
  - The device’s encrypted private key (e.g., device.des3.key)
  - The device certificate (e.g., device.crt)

Procedure

Step 1 Enter global configuration mode.

Step 2 Import the root CA certificate using the `crypto pki import [trustpoint] pem terminal password [password]` command.
When prompted, paste the contents of your rootCA.pem file, followed by "quit" on a new line.

Step 3 When prompted, import the device's encrypted private key.

a) Paste contents, a by "quit", as instructed.

Step 4 When prompted, import the device certificate.

a) Paste contents, followed by "quit," as instructed.

Step 5 Configure trustpoint parameters as needed (such as revocation checks).

Example:

```
# Send:
Device# configure terminal
Device(config)# crypto pki import trustpoint1 pem terminal password password1

# Receive:
% Enter PEM-formatted CA certificate.
% End with a blank line or "quit" on a line by itself.

# Send:
# Contents of rootCA.pem, followed by newline + 'quit' + newline:
-----BEGIN CERTIFICATE-----
<snip>
-----END CERTIFICATE-----
quit

# Receive:
% Enter PEM-formatted encrypted private General Purpose key.
% End with "quit" on a line by itself.

# Send:
# Contents of device.des3.key, followed by newline + 'quit' + newline:
-----BEGIN RSA PRIVATE KEY-----
Proc-Type: 4,ENCRYPTED
DEK-Info: DES-EDE3-CBC,D954FF9E43F1BA20
<snip>
-----END RSA PRIVATE KEY-----
quit

# Receive:
% Enter PEM-formatted General Purpose certificate.
% End with a blank line or "quit" on a line by itself.

# Send:
# Contents of device.crt, followed by newline + 'quit' + newline:
-----BEGIN CERTIFICATE-----
<snip>
-----END CERTIFICATE-----
quit

# Receive:
% PEM files import succeeded.
Device(config)#

# Send:
Device(config)# crypto pki trustpoint trustpoint1
Device(ca-trustpoint)# revocation-check none
Device(ca-trustpoint)# end
Device#
```

The certificates and key are installed, enabling secure gNMI server operation on Cisco IOS XE.

Collection job status fields and interpretations

Collection job status provides detailed information regarding each active collection job across all Embedded Collector instances registered to Cisco Crosswork.

View active collection jobs

You can monitor the status of collection jobs that are currently active on all Embedded Collector instances enrolled with Crosswork Network Controller from the Collection Jobs page. The left pane lists all active collection jobs and displays their Status, App ID, Context ID, and Action. Use the Action drop-down to delete collection jobs or refresh the status of a collection job and its associated tasks.

The **Job Details** pane shows the details of all collection tasks that are associated with a particular job in the left pane. The overall status of the Collection job in the **Collection Jobs** pane is the aggregate status of all the collection tasks in the **Jobs Details** pane.

View job details

When you select a job in the Collection Jobs pane, the following details are displayed in the Job Details pane:

- Application name and context associated with the collection job
- Collection job status
- Job configuration of the collection job that you pass in the REST API request. Click the icon next to Config Details to view the job configuration. Crosswork Network Controller lets you view configuration in two modes:
  - View Mode
  - Text Mode
- Collection type
- Last modified date and time
- Collections (x): Number of input collections; Issues (y) shows how many are UNKNOWN or FAILED
- Distributions (x): Number of output collections; Issues (y) shows how many are UNKNOWN or FAILED

The overall job status is based on the aggregate status of all related collection tasks.

Key parameters shown for collection jobs

The Collection Jobs and Job Details panes display these information for active collection jobs:

Table 9: Collection jobs fields and descriptions

Field	Description
Collection/Distribution Status	Current status of the collection/distribution; updated on change. Click icon for details.

Field	Description
Hostname	Device hostname for the collection job.
Device Id	Unique identifier of the device being monitored.
Sensor Data	The sensor paths involved; view metrics summary for cadence-based statistics. Click  to see collection/distribution summary. From the sensor data summary pop-up you can copy the sensor data by clicking Copy to Clipboard. Click  to see collection/distribution metrics summary. The metrics are reported on a cadence-basis, i.e., once every 10 minutes by default. It shows the following metrics for a collection: • last_collection_time_msec • total_collection_message_count • last_device_latency_msec • last_collection_cadence_msec It shows the following metrics for a collection: • total_output_message_count • last_destination_latency_msec • last_output_cadence_msec • last_output_time_msec • total_output_bytes_count
Destination	Data destination for the job.
Last Status Change Reported Time	Time and date of the last status change was reported for the device sensor pair.

Guidelines for interpreting the collection job states

- The status of a collection task associated with a device after it is attached to an Embedded Collector, is **Unknown**.
- A job could have status as **Unknown** for one of the following reasons:
  - Embedded Collectors have not yet reported its status.
  - Loss of connection between Embedded Collectors and Cisco Crosswork.
  - Embedded Collectors have received the collection job, but the actual collection is still pending. For example, traps are not being sent to Embedded Collectors southbound interface, or the device is not sending telemetry updates.

- The trap condition in an SNMP trap collection job which we are monitoring has not occurred. For example, if you are looking for Link Up or Link down transitions and the link state has not changed since the collector was established, then the state reports as **Unknown**. To validate that trap-based collections are working, it is therefore necessary to actually trigger the trap.

- After the collection job is processed, the status changes to 'Successful' if the processing was successful or else it changes to 'Failed'.
- If a collection job is in a degraded state, one of the reasons might be that the static routes to the device have been erased from Embedded Collectors.
- Collections to a destination that is in an Error state do not stop. The destination state is identified in the background. If the destination is in an Error state, the error count is incremented. Drill down on the error message that is displayed in the **Distribution** status to identify and resolve the issue by looking at respective collector logs.
- Cisco Crosswork Health Insights - KPI jobs must be enabled only on devices mapped to an extended Embedded Collector instance. Enabling KPI jobs on devices that are mapped to a standard Embedded Collector instance reports the collection job status as **Degraded** and the collection task status as **Failed** in the **Jobs Details** pane.

Create a collection job

You can publish collection jobs created through the Crosswork Network Controller UI page only once.

Use the Crosswork Network Controller UI to create jobs that collect device or network data via CLI or SNMP. These jobs enable centralized data collection for analysis or export.

Before you begin

Ensure that a data destination is created and active for depositing collected data. Make sure you have the details of the sensor path and MIB from which you plan to collect data.

Procedure

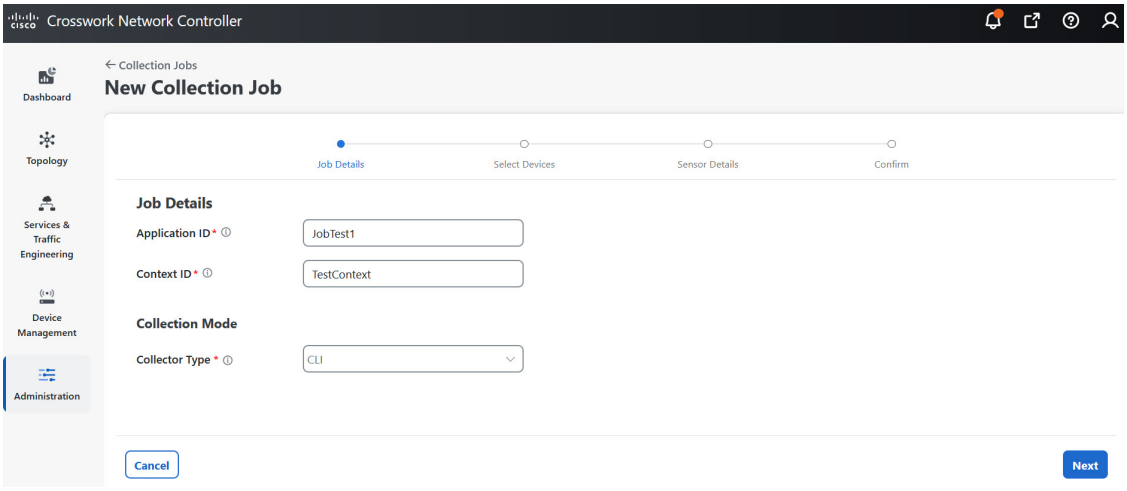
Step 1 Go to **Administration > Data Gateway Management > Collection Jobs > Bulk Jobs**.

Step 2 Click  .

Step 3 On the **Job details** page, enter values for these fields:

- **Application ID**: A unique identifier for the application.
- **Context**: A unique identifier to identify your application subscription across all collection jobs.
- **Collector Type**: Select the type of collection as either CLI or SNMP.

Figure 7: Job details



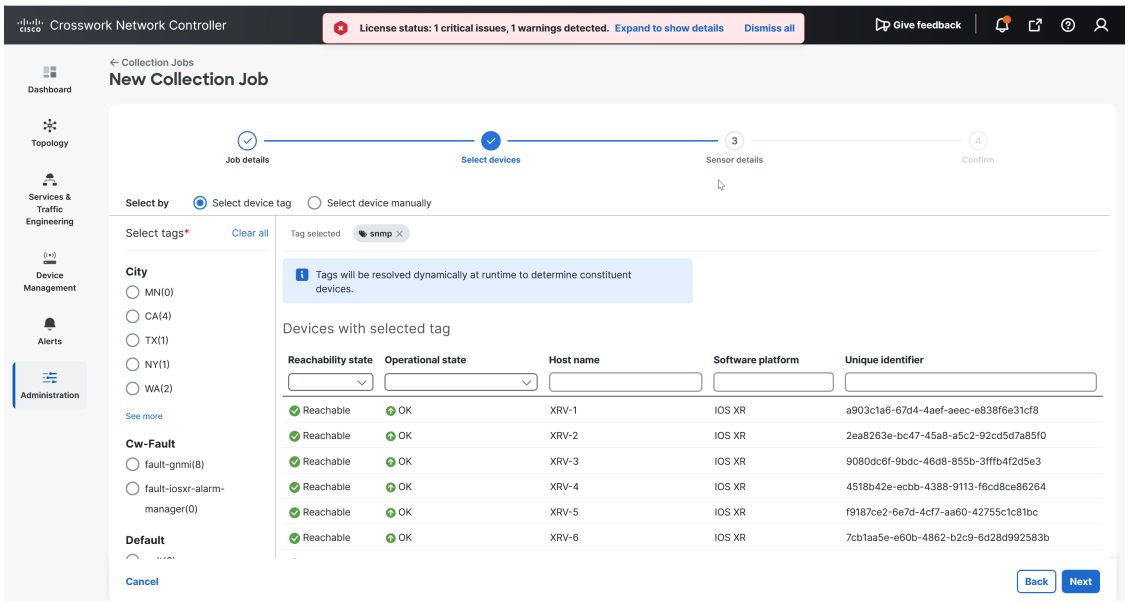
a) Click **Next**.

Step 4 Select the devices from which you want to collect data.

You can select devices by device tag or choose them manually.

a) Click **Next**.

Figure 8: Select devices



Step 5 Enter these sensor details for CLI collection:

a) Select data destination from the **Select Data Destination** drop-down list.

b) Select sensor type from **Sensor Types** pane on the left.

c) If you select **CLI PATH**, click + and enter these parameters in the **Add CLI Path** dialog box:

- **Collection Cadence**: Push or poll cadence in seconds

- **Command**: CLI command
- **Topic**: Topic associated with the output destination

Note
Topic can be any string if using an external gRPC server.

d) If you select **Device Package**, click + and enter these parameters in the **Add Device Package Sensor** dialog box:

- **Collection cadence**: Push or poll cadence in seconds
- **Device Package Name**: Custom XDE device package ID used when creating the device package
- **Function name**: Function name within custom XDE device package
- **Topic**: Topic associated with the output destination

e) Enter Key and String value for the parameters.
f) Click **Save**.

Figure 9: Sensor details for CLI path

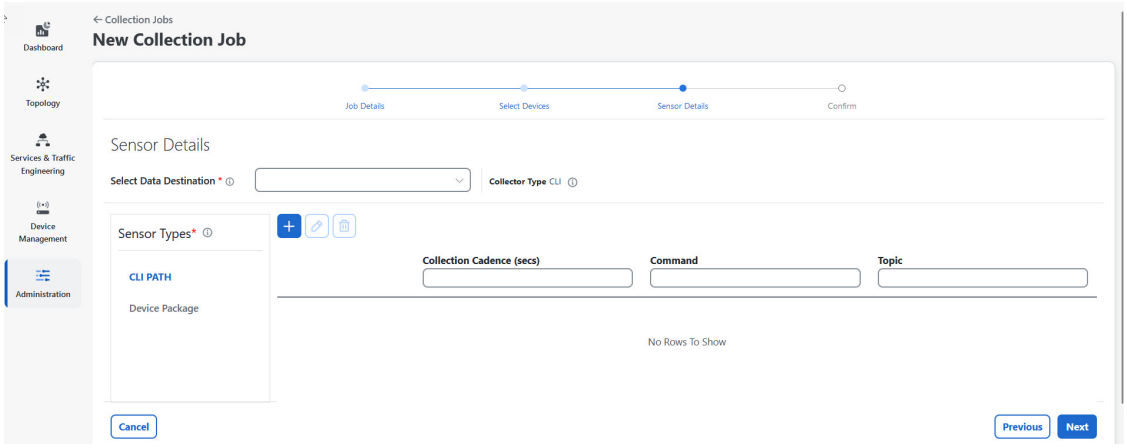


Figure 10: Add CLI path

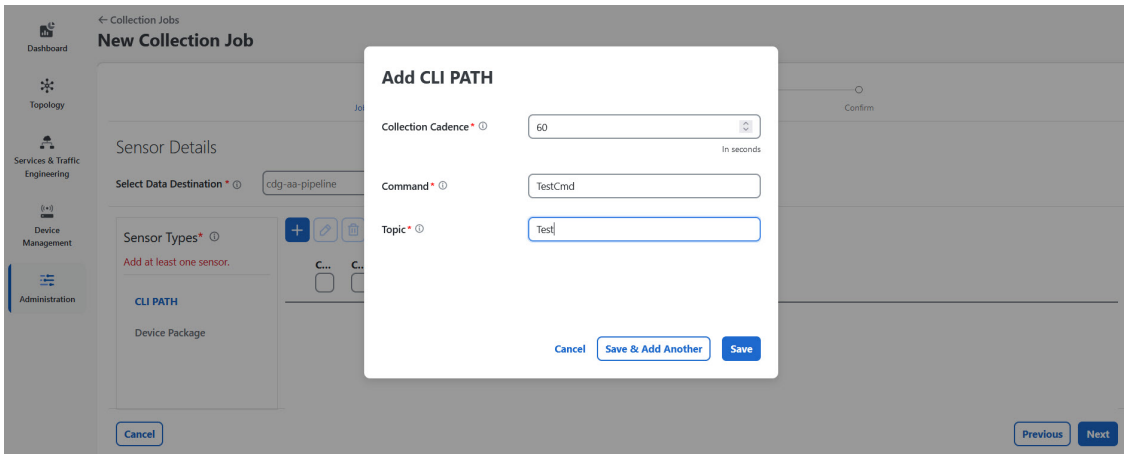
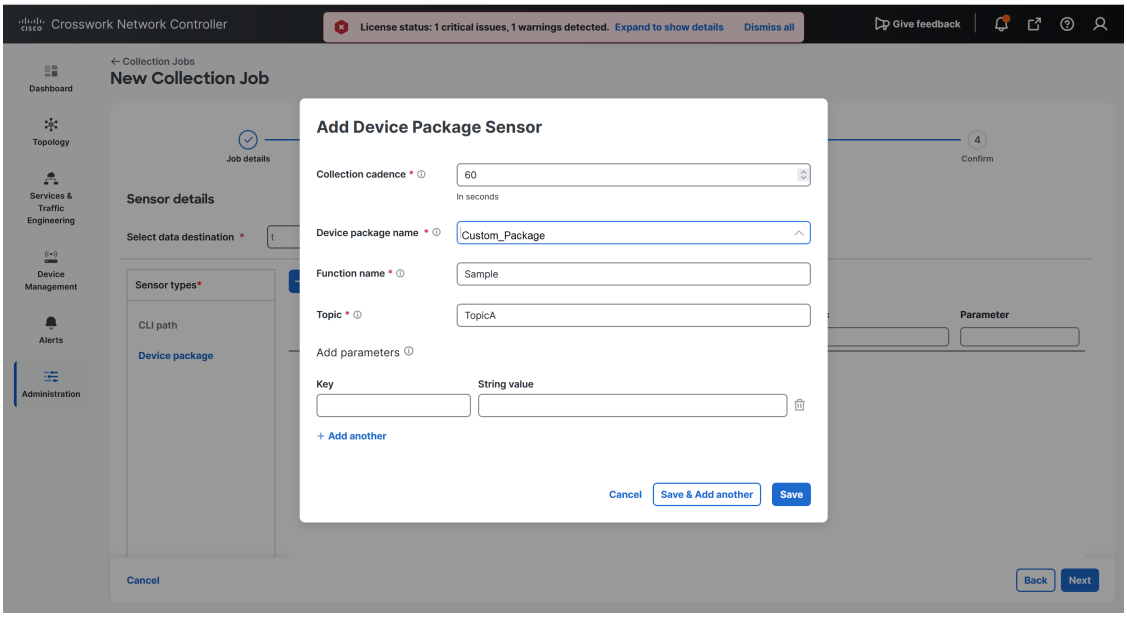


Figure 11: Add device package sensor



Step 6 Enter these sensor details for SNMP collection:

a) Select data destination from the **Select Data Destination** drop-down list.

b) Select sensor type from **Sensor Types** pane on the left.

c) If you select **SNMP MIB**, click + and enter these parameters in the **Add Device Package Sensor** dialog box:

- **Collection Cadence**: Push or poll cadence in seconds.
- **OID**
- **Operation**: Select the operation from the list.
- **Topic**: Topic associated with the output destination.

d) If you select **Device Package**, click + and enter values for these parameters in the Add Device Package Sensor dialog box:

- **Collection Cadence**: Push or poll cadence in seconds.
- **Device Package Name**: Custom device package ID used when creating the device package.
- **Function name**: Function name within custom device package.
- **Topic**: Topic associated with the output destination.

e) Enter the key and string value for the parameters.

f) Click **Save**.

Figure 12: Sensor details for SNMP path

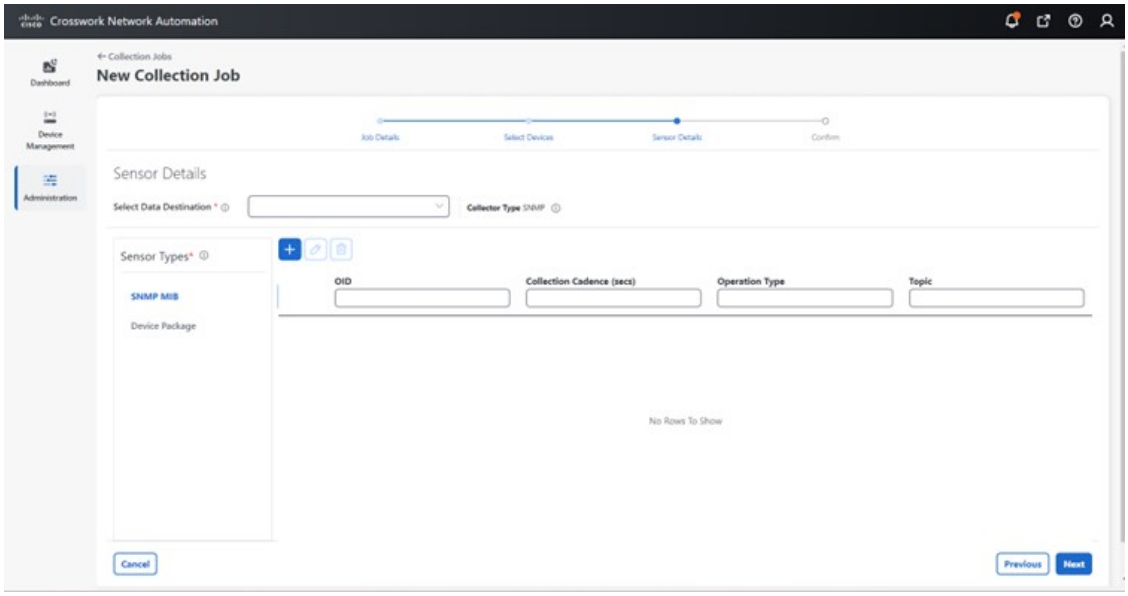
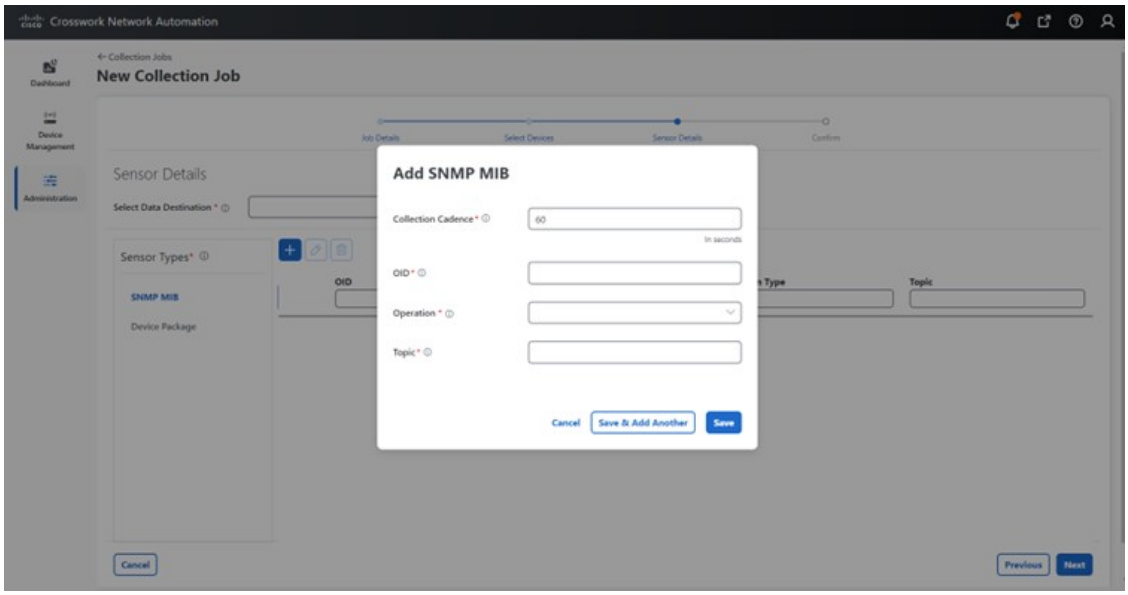


Figure 13: Add SNMP MIB



Step 7 Click **Create Collection Job**.

The new collection job is published and starts gathering data according to configuration.

What to do next

Review job status and ensure data is being deposited to the specified destination as expected.

View active collection jobs

View all currently active collection jobs on Embedded Collector instances.

Use the **Collection Jobs** page in Cisco Crosswork to get an overview of every active collection job and interact with status and management controls.

Procedure

Step 1 Go to the **Collection Jobs** page in the Crosswork Network Controller UI.

Step 2 From the left navigation bar, choose **Administration > Collection Jobs**.

Step 3 Review the list of active collection jobs and note their Status, App ID, Context ID, and Action.

Step 4 Use the **Action** drop-down to delete a collection job or refresh the status of a job and its associated tasks.

You can view the real-time status of active collection jobs and manage them for all enrolled Embedded Collector instances.

Delete a collection job

Remove an unwanted external collection job from your system.

Deleting a collection job also deletes its associated collection tasks. Delete jobs created by Health Insights by disabling the KPI profile. This action removes the collection jobs that the profile deployed.

Before you begin

Do not delete system jobs created by Crosswork Applications as deletion could cause collection issues.

Jobs created by Health Insights should only be deleted by disabling the KPI profile which will remove the collection jobs it deployed.

Procedure

Step 1 Go to **Administration > Collection Jobs**.

Step 2 Select the **Bulk Jobs** or **Parametrized Jobs** tab.

Step 3 Select the collection job you want to delete.

Step 4 In the job's row, select **Delete**.

Step 5 Confirm by clicking **Delete** in the confirmation window.

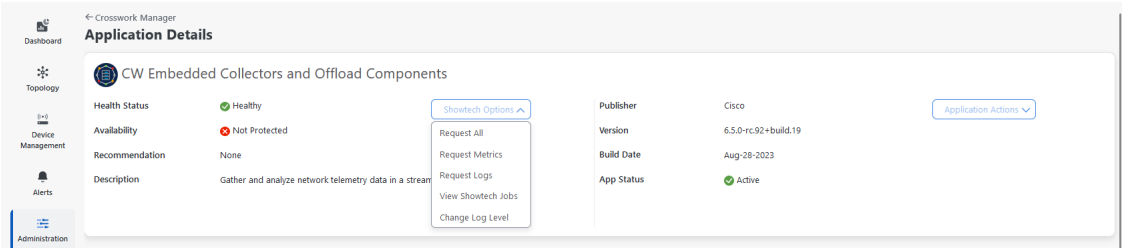
The selected collection job and its collection tasks are deleted.

Check the health status of Embedded Collectors

View the operational health and status of all embedded collectors in Cisco Crosswork.

The CW Embedded Collectors and Offload Components tile provides the operations and health summary of Embedded Collectors. You can find information about the health of pods running on the Crosswork container on this page. The overall health of the Embedded Collectors application depends on the health of each individual pod service.

Figure 14: Crosswork Embedded Collector and Offload Component



Procedure

Step 1 From the main menu, choose **Administration > Crosswork Manager > CW Embedded Collectors and Offload Components**.
You can see the health and status summary of embedded collectors, along with detailed pod health information, in the displayed pane.

Step 2 Optionally, use the **Showtech Options** drop-down to:

- **Request All**: Collects both logs and metrics for the embedded collectors. To view the logs, navigate to Crosswork Manager > Application Management > Showtech Requests.
- **Request Metrics**: Collects only metrics information.
- **Request Logs**: Collects only log information.
- **View Showtech Jobs**: Displays the progress and status of Showtech jobs. You can also see job details from Crosswork Manager > Application Management > Showtech Requests.
- **Change Log Level**: Changes the log level of selected Embedded Collector components, such as collectors (e.g., cli-collector) and infrastructure services (e.g., oam-manager). The change affects only the targeted collector.

Step 3 Optionally, use the **Application Actions** drop-down to:

- **Install**: Installs a new collector with profile, hostname, and management interface details from previous collectors.
- **Upgrade**: Upgrades the collector to a newer version.
- **Activate**: Activates selected collectors.
- **Uninstall**: Removes the collector application. Uninstalling the application may interrupt ongoing collection jobs and cause data loss for current operations.

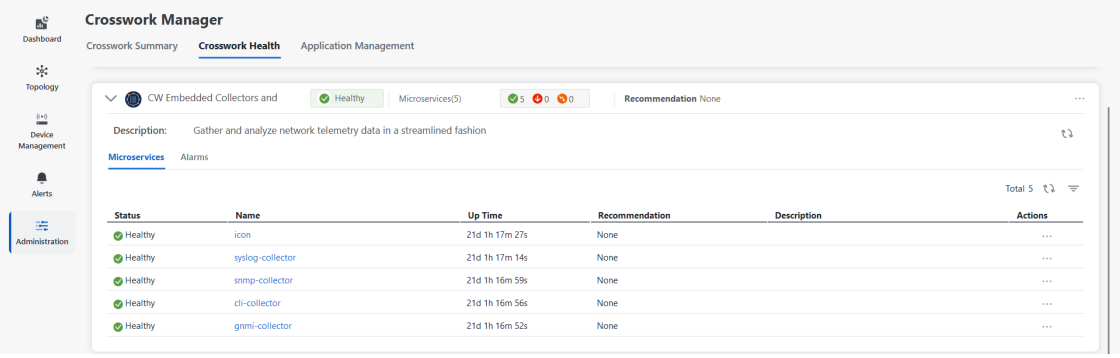
You have reviewed the current health status, accessed logs and metrics as needed, and can take corrective actions.

Monitor the collector's pod health

Ensure the health of the collector pod to maintain optimal collection performance. Take timely corrective action to prevent service disruption.

In the Embedded Collector and Offload Component pane, you can view a detailed overview of the health status of pods hosting collectors or microservices. We recommend regularly monitoring the health of the collector pods in your network. This helps avoid overloading and enables proactive corrective measures, such as adding more resources or reducing the load on the collector in a timely manner.

Figure 15: Microservices Tab



Procedure

Step 1 Go to **Administration > Crosswork Manager > CW Embedded Collectors and Offload Components**.

Step 2 Expand CW Embedded Collectors and Offload Components, then select **Microservices**.

Step 3 (Optional) In the **Microservices** tab, type the collector name in the **Name** field to locate the collector pod.

Step 4 (Optional) From this page, click  under the **Actions** column to perform these actions:

- **Restart**: restarts the collector pod. Restarting a pod disrupts the ongoing collection process. If you need to restart, start, or stop a process, we strongly advise consulting the Cisco TAC team.
- **Showtech Requests**: displays the showtech jobs executed for the corresponding collector pod. To view the logs, go to **Crosswork Manager > Application Management > Showtech Requests**.
- **Request All**: collects both logs and metrics of the pod. To view the logs, go to **Crosswork Manager > Application Management > Showtech Requests.**
- **Request Metrics**: collects the metrics of the pod.
- **Request Logs**: collects the logs of the pod.

View collector alarms and events

Enable users to monitor collector alarms and events to identify and remediate potential data collection issues.

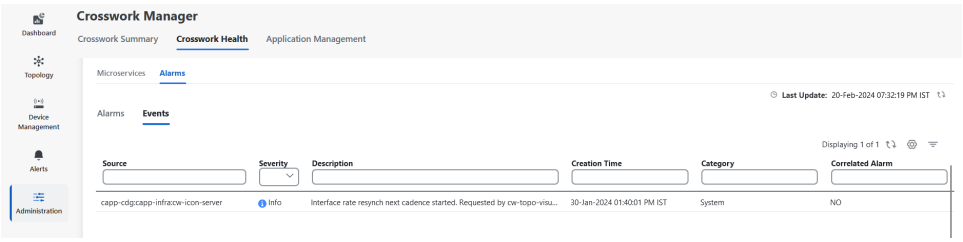
Embedded Collectors in the Crosswork Network Controller generate alarms when anomalies prevent data collection. By monitoring these alarms, you can detect issues that affect collector performance and take necessary action.

Procedure

Step 1 From the Crosswork Network Controller UI, go to **Administration > Crosswork Manager > CW Embedded Collectors and Offload Components**.

The **Alarms** tab provides a consolidated list of all collector alerts and events.

Figure 16: Events



Step 2 Toggle between the **Alarms** and **Events** subtabs to view the respective details.

Step 3 Filter the alarms or events list by:

- adjusting column filters,
- changing the **Active Alarms Only** slider, and
- adding or removing columns using the ⚙ icon.

You can view all current collector alarms and events. This enables timely detection and troubleshooting of potential data collection issues.

Embedded Collector troubleshooting scenarios

The troubleshooting section contains information about the possible issues and corrective actions that you may observe with Embedded Collectors.

Troubleshooting the admin state change from DOWN to UP

In a single VM deployment, devices are automatically attached to Embedded Collectors, which causes their Admin State to change from DOWN to UP.

Workaround: If you need to change the Admin State to DOWN, manually update it using the **Edit Device** page. For detailed steps on editing a device, see the **Edit Devices** section in the *Cisco Crosswork Network Controller 7.2 Device Lifecycle Management* guide.

Kafka connection failure for unsecured external destinations

If a collection job is submitted to an external (unsecured) Kafka destination, the dispatch job may fail to connect to the Kafka server.

Collector error:

```
org.apache.kafka.common.errors.TimeoutException: Topic cli-job-kafka-unsecure not present
in metadata after 60000 ms.
```

Kafka error:

```
SSL authentication error "[2021-01-08 22:17:03,049] INFO [SocketServer brokerId=0] Failed
authentication with /80.80.80.108 (SSL handshake failed)
(org.apache.kafka.common.network.Selector).
```

These errors appear when the required port on the external Kafka VM is blocked, preventing the collector from connecting.

Workaround:

1. On the Kafka Docker host or server, run this command to verify whether the Kafka port is listening:

   ```
   netstat -tulpn
   ```
2. Unblock or correct the port configuration on the Kafka server. After fixing the issue, restart the Kafka server process to restore connectivity.