



Cisco Crosswork Data Gateway 1.0 User Guide

First Published: 2019-07-29

Americas Headquarters

Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134-1706
USA
<http://www.cisco.com>
Tel: 408 526-4000
800 553-NETS (6387)
Fax: 408 527-0883

THE SPECIFICATIONS AND INFORMATION REGARDING THE PRODUCTS IN THIS MANUAL ARE SUBJECT TO CHANGE WITHOUT NOTICE. ALL STATEMENTS, INFORMATION, AND RECOMMENDATIONS IN THIS MANUAL ARE BELIEVED TO BE ACCURATE BUT ARE PRESENTED WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED. USERS MUST TAKE FULL RESPONSIBILITY FOR THEIR APPLICATION OF ANY PRODUCTS.

THE SOFTWARE LICENSE AND LIMITED WARRANTY FOR THE ACCOMPANYING PRODUCT ARE SET FORTH IN THE INFORMATION PACKET THAT SHIPPED WITH THE PRODUCT AND ARE INCORPORATED HEREIN BY THIS REFERENCE. IF YOU ARE UNABLE TO LOCATE THE SOFTWARE LICENSE OR LIMITED WARRANTY, CONTACT YOUR CISCO REPRESENTATIVE FOR A COPY.

The Cisco implementation of TCP header compression is an adaptation of a program developed by the University of California, Berkeley (UCB) as part of UCB's public domain version of the UNIX operating system. All rights reserved. Copyright © 1981, Regents of the University of California.

NOTWITHSTANDING ANY OTHER WARRANTY HEREIN, ALL DOCUMENT FILES AND SOFTWARE OF THESE SUPPLIERS ARE PROVIDED "AS IS" WITH ALL FAULTS. CISCO AND THE ABOVE-NAMED SUPPLIERS DISCLAIM ALL WARRANTIES, EXPRESSED OR IMPLIED, INCLUDING, WITHOUT LIMITATION, THOSE OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NON-INFRINGEMENT OR ARISING FROM A COURSE OF DEALING, USAGE, OR TRADE PRACTICE.

IN NO EVENT SHALL CISCO OR ITS SUPPLIERS BE LIABLE FOR ANY INDIRECT, SPECIAL, CONSEQUENTIAL, OR INCIDENTAL DAMAGES, INCLUDING, WITHOUT LIMITATION, LOST PROFITS OR LOSS OR DAMAGE TO DATA ARISING OUT OF THE USE OR INABILITY TO USE THIS MANUAL, EVEN IF CISCO OR ITS SUPPLIERS HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

Any Internet Protocol (IP) addresses and phone numbers used in this document are not intended to be actual addresses and phone numbers. Any examples, command display output, network topology diagrams, and other figures included in the document are shown for illustrative purposes only. Any use of actual IP addresses or phone numbers in illustrative content is unintentional and coincidental.

All printed copies and duplicate soft copies of this document are considered uncontrolled. See the current online version for the latest version.

Cisco has more than 200 offices worldwide. Addresses and phone numbers are listed on the Cisco website at www.cisco.com/go/offices.

Cisco and the Cisco logo are trademarks or registered trademarks of Cisco and/or its affiliates in the U.S. and other countries. To view a list of Cisco trademarks, go to this URL: [www.cisco.com go trademarks](http://www.cisco.com/go/trademarks). Third-party trademarks mentioned are the property of their respective owners. The use of the word partner does not imply a partnership relationship between Cisco and any other company. (1721R)

© 2019 Cisco Systems, Inc. All rights reserved.



CONTENTS

CHAPTER 1

Overview 1

- Audience 1
- Overview 1
- Log In and Log Out 2
 - Access Crosswork Data Gateway Through vCenter 2
 - Access Crosswork Data Gateway Via SSH 3
- Log Out 4
- Use the Interactive Console 5

CHAPTER 2

Get Started 7

- Before You Begin 7
- Basic Concepts 8
 - Key Elements of Crosswork Data Gateway Solution 8
 - Crosswork Data Gateway Components 8
 - Controller Gateway 8
 - Image Manager 9
 - Vitals Monitor 9
 - Route Manager 9
 - Docker Engine 9
 - Crosswork Data Gateway Configuration Files 9
 - boot-config 10
 - docker-compose 12
 - Enrollment Package 14
 - Static-routes Config 17
- Crosswork Data Gateway Solutions 17
 - Crosswork Data Gateway for Crosswork Cloud 18

Crosswork Data Gateway for Customer-hosted Solutions 19
 High-Level Workflow 22

CHAPTER 3 **Set Up Crosswork Data Gateway 23**

Set Up Controller 23
 Set Up Output Channels 23
 Output Data Format 24
 Crosswork Data Gateway Authentication and Bootstrap 26

CHAPTER 4 **Manage Users 29**

Supported User Roles 29
 Change Password 31

CHAPTER 5 **Configure Crosswork Data Gateway 33**

View Current System Settings 33
 View Management and South/North-bound Addresses 35
 View NTP Settings 35
 View DNS Settings 35
 View Proxy Settings 35
 View VM UUID 35
 View Syslogs 36
 View Certificates 36
 Change Current System Settings 36
 Configure NTP 38
 Configure DNS 39
 Configure Control Proxy 39
 Configure Static Routes 39
 Add Static Routes 39
 Delete Static Routes 41
 Configure Syslog 42
 Create New SSH Keys 43
 Import Certificate 43

CHAPTER 6 **Collection Jobs 45**

About Collection Jobs	45
Collection Job Payload Model	46
Add/Delete Collection Jobs	53
Best Practices and Limitations for Creating Collection Jobs	54
Sample Collection Jobs Payloads	55
CLI Collection Job	55
SNMP Collection Jobs	65
SNMP Collection Job	68
SNMP Traps Collection Job	78
MDT Collection Job	84
MDT gRPC Dial-in Mode	85
MDT gRPC Dial-out Mode	86
MDT TCP Dial-out Mode	87

CHAPTER 7	Deploying Custom Packages	89
	Types of Custom Packages	89
	Deploy a Custom Device Package	89
	Deploy a Custom MIB	90

CHAPTER 8	Monitor Crosswork Data Gateway Health	93
	Vitals Monitor	93
	View Crosswork Data Gateway Vitals	93
	View Docker Containers Vitals	94
	View Docker Images Vitals	95
	View Controller Reachability Vitals	95
	View NTP Reachability Vitals	96
	View Route Table	96
	View ARP Table	96
	View Network Connections	96
	View Disk Space Usage	97
	collector-vitals Service	97

CHAPTER 9	Troubleshooting	101
	Troubleshoot Crosswork Data Gateway	101

[Ping a Host](#) 103

[Traceroute to a Host](#) 104

[Check NTP Status](#) 105

[Check System Uptime](#) 106

[Run show-tech](#) 106

[Reboot Crosswork Data Gateway VM](#) 107

[Start/Stop Docker Containers](#) 108



CHAPTER 1

Overview

This section contains the following topics:

- [Audience, on page 1](#)
- [Overview, on page 1](#)
- [Log In and Log Out, on page 2](#)
- [Use the Interactive Console, on page 5](#)

Audience

This guide is for experienced network administrators who want to use Cisco Crosswork Data Gateway in their network. This guide assumes that you are familiar with the following topics:

- Linux system administration
- Networking technologies and protocols
- Familiarity with the different operating systems used on devices that form your network, such as Cisco IOS-XR, ISO-XE, and NX-OS.

Overview

Cisco Crosswork Data Gateway is a model-driven data collection platform that enables real-time data collection from multi-protocol capable devices, thereby reducing the need for multiple collection points to the network.

Crosswork Data Gateway provides a simple secure gateway between the network and applications to collect data. It enables service providers to quickly program and deploy collectors and offers central visibility into services collecting data and type of data being collected.

Crosswork Data Gateway is available in two variants to enable data collection in different use cases:

- **Cisco Crosswork Cloud:** Provides secure data collection and distribution to Cisco Crosswork Cloud solutions, such as Trust Insights, Network Insights, etc.
- **Customer-hosted Solutions:** Crosswork Data Gateway provides data collection and distribution to customer-hosted solutions.

Log In and Log Out

You can use either of the following two ways to access Crosswork Data Gateway:

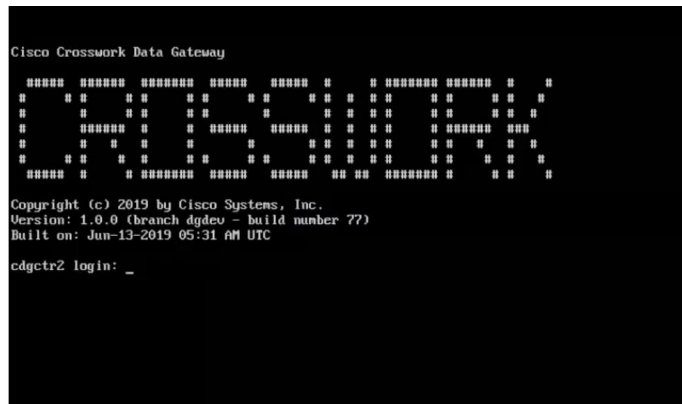
- [Access Crosswork Data Gateway Through vCenter, on page 2](#)
- [Access Crosswork Data Gateway Via SSH, on page 3](#)

Access Crosswork Data Gateway Through vCenter

Follow these steps to login via vCenter:

Step 1 Locate the VM in vCenter and then right click and select **Open Console**.

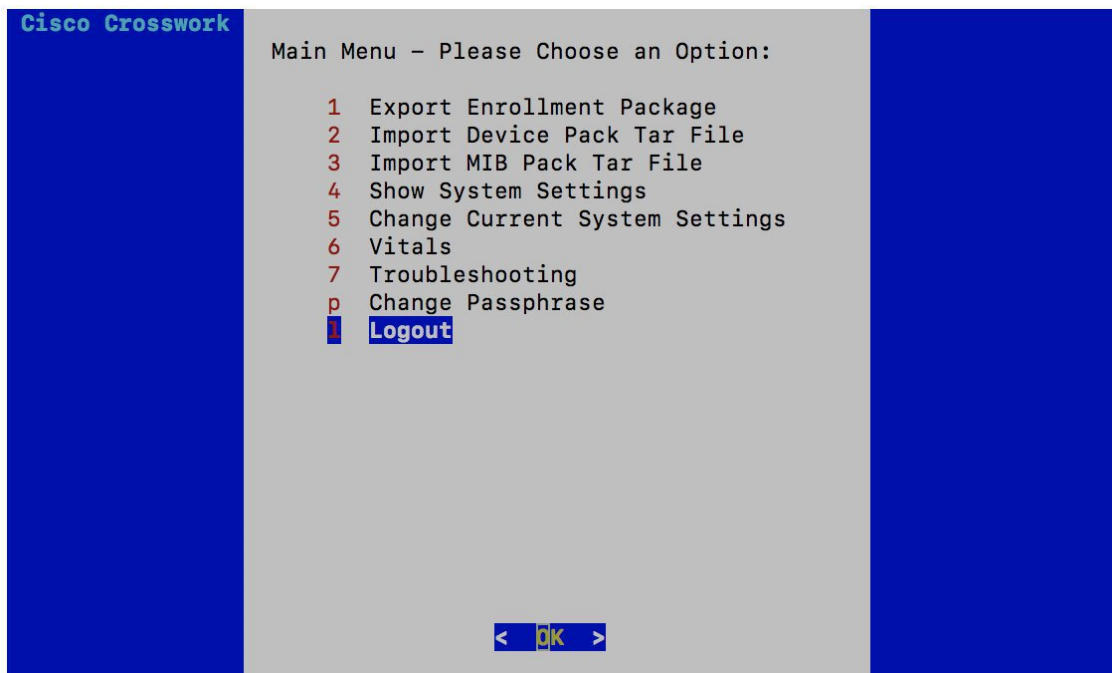
The Crosswork Data Gateway flash screen comes up, prompting for password:



Step 2 Input the password and press Enter.

Upon successful login, you can see the Crosswork Data Gateway's interactive console, as shown in the following figure:

Note The Main Menu shown in this figure corresponds to **dg-admin** user. It is different for **dg-oper**. See *Chapter 3 Manage Users in Crosswork Data Gateway 1.0 User Guide*.



Access Crosswork Data Gateway Via SSH

Follow these steps to login via SSH.

Step 1

Run the following command:

```
ssh <username>@<ManagementNetworkIP>
```

where **ManagementNetworkIP** is the management network IP address.

For example,

To login as administrator user: `ssh dg-admin@<ManagementNetworkIP>`

To login as operator user: `ssh dg-oper@<ManagementNetworkIP>`

The following Crosswork Data Gateway flash screen opens prompting for password:

Use the Interactive Console

Crosswork Data Gateway launches an interactive console upon successful login. The interactive console displays the Main Menu as shown in the following figure:



Note The Main Menu shown here corresponds to **dg-admin** user. It is different for **dg-oper** user as the operator does not have same privileges as the administrator. See [Supported User Roles, on page 29](#).

```
Cisco Crosswork
Main Menu - Please Choose an Option:

1 Export Enrollment Package
2 Import Device Pack Tar File
3 Import MIB Pack Tar File
4 Show System Settings
5 Change Current System Settings
6 Vitals
7 Troubleshooting
p Change Passphrase
! Logout

< OK >
```

The Main Menu presents the following options:

1. Export Enrollment Package
2. Import Device Pack Tar File
3. Import MIB Pack Tar File
4. Show System Settings
5. Change Current System Settings
6. Vitals
7. Troubleshooting
8. Change Passphrase
9. Logout



CHAPTER 2

Get Started

This section contains the following topics:

- [Before You Begin, on page 7](#)
- [Basic Concepts, on page 8](#)
- [Crosswork Data Gateway Solutions, on page 17](#)
- [High-Level Workflow, on page 22](#)

Before You Begin



Note This section is applicable only for customer-hosted solutions.

Before you begin using Crosswork Data Gateway, Cisco recommends that you complete the following planning and information-gathering steps, in any order you wish:

- **User Roles:** Cisco recommends that you use role-based access control to confine users to just the software functions needed to perform their job duties. See [Supported User Roles, on page 29](#).
- **Data Collection:** Decide what type of data collection method (CLI, SNMP, or MDT) you want to use, the type of data you want to collect, and duration for which you want to collect the data.
- **Devices:** If you are using Crosswork Data Gateway for customer-hosted solution, decide the number of devices you want to collect data from, types of the devices, and OS running on the devices.



Note Crosswork Data Gateway is read only to the network. The devices must be configured for the correct data metrics in advance of the Crosswork Data Gateway collecting them.

- **Output Destination:** Decide which output server (Kafka or gRPC) you are going to use and ensure it is set up to receive input from Crosswork Data Gateway.
- **Controller:** Have your Controller ready to be integrated with Crosswork Data Gateway.

Basic Concepts

Cisco Crosswork Data Gateway makes extensive use of certain concepts. It is helpful to be familiar with them before you get started.

Key Elements of Crosswork Data Gateway Solution

Depending on its role, Crosswork Data Gateway could either be used as a collector to corral data from several devices or it could be used as a gateway for network devices to reach Cloud applications.

Whatever the case may be, every Crosswork Data Gateway solution has the following two key elements:

1. **Controller:** Controls the collection of data and its distribution to output destinations. It also instructs the Crosswork Data Gateway where it can download functional images from. A Crosswork Data Gateway instance can have only one Controller.
2. **Crosswork Data Gateway:** Follows the instructions of the Controller - collects data as requested and sends it to the defined output destination. It consists of the following:
 - A Base VM containing an operating system and basic functionalities
 - Additional functional images to enable protocol-based data collection.

Crosswork Data Gateway Components

Crosswork Data Gateway has the following five main components or SystemD services:

- [Controller Gateway, on page 8](#)
- [Image Manager, on page 9](#)
- [Vitals Monitor, on page 9](#)
- [Route Manager, on page 9](#)
- [Docker Engine, on page 9](#)

Controller Gateway

Controller Gateway is the component responsible for all the interaction between a Crosswork Data Gateway instance and its Controller. It manages the session creation with Controller and makes sure all the payloads and responses are signed and verified for integrity. Components such as Image Manager, Vitals Monitor, Route manager and all collectors interact via Controller Gateway with Controller to exchange the details those components need.

**Note**

When the Controller Gateway stops, any alerts are not updated in `cdg-alerts.log`. However, when it starts, it sends an alert that it has started. This is because all the alerts go through the Controller Gateway and if it is down, Controller won't receive the alerts.

Image Manager

The Image Manager comes into play upon Crosswork Data Gateway VM boot up. It establishes a session with the Controller using information from the [Enrollment Package, on page 14](#) generated during VM bootstrap. It then downloads the functional images from the repository as instructed by the Controller and brings up the containers.

It has the following responsibilities:

- Periodically pull [boot-config, on page 10](#) file from the Controller via Controller Gateway.
- Based on the boot-config and local images metadata cache, determine if the functional images and [docker-compose, on page 12](#) file need to be downloaded.
- Send appropriate alerts to Controller, if there are issues while processing the boot-config.
- Stop and remove any containers that are no longer called for in the latest boot-config.
- Cleanup the local images metadata cache to keep it synchronized with the latest boot-config received from Controller.

Vitals Monitor

The Vitals Monitor monitors the health and vitals of the Crosswork Data Gateway VM. It samples the CPU, memory, disk usage, docker containers details, etc. and aggregates this information in a file on the host filesystem.

In Cloud solution, the host is mounted on downloaded functional image container app. The app may then collect relevant metrics and send to Controller for further processing.

However, in a customer-hosted solution, the vitals are periodically archived in a local persistent volume that is mounted within each container. This information is accessible from both VM and the Docker container. This data can be exported to the Controller periodically.

For more information, see [Monitor Crosswork Data Gateway Health, on page 93](#).

Route Manager

Route Manager periodically syncs the configured static routes from the Controller. It then directs the traffic to devices on different Southbound destinations based on the routes.

Route manager adds/deletes the static routes by comparing the existing routes configured on the VM with the routes configuration pulled from the Controller.

Appropriate alerts are sent to the Controller if there is any failure in processing routes configuration.

Docker Engine

The Docker Engine manages all containers.

Crosswork Data Gateway Configuration Files

Crosswork has the following three important configuration files:

- [boot-config, on page 10](#)
- [docker-compose, on page 12](#)

- [Enrollment Package, on page 14](#)

boot-config

Once the session with the Controller is established, the Image Manager requests the Controller for a `boot-config` response. The Controller prepares a `boot-config` response containing a list of functional images, docker-compose file, their SHA digests, and the location from where the image tar files can be downloaded.

The following figure shows a sample `boot-config` response:

```
{
  "pollingPeriodSecs": 30,
  "files": [
    {
      "fileType": "CONTAINER_FILE",
      "url": "<url_of_container_file>",
      "hash": "<hash>",
      "container": {
        "name": "mdt-collector",
        "imageRepoName": "<image_repo_name>",
        "imageRepoTag": "<image_repo_tag>"
      }
    },
    {
      "fileType": "CONTAINER_FILE",
      "url": "<url_of_container_file>",
      "hash": "<hash>",
      "container": {
        "name": "cli-collector",
        "imageRepoName": "<image_repo_name>",
        "imageRepoTag": "<image_repo_tag>"
      }
    },
    {
      "fileType": "CONTAINER_FILE",
      "url": "<url_of_container_file>",
      "hash": "<hash>",
      "container": {
        "name": "snmp-collector",
        "imageRepoName": "<image_repo_name>",
        "imageRepoTag": "<image_repo_tag>"
      }
    },
    {
      "fileType": "COMPOSE_FILE",
      "url": "url_of_compose_file",
      "hash": "<hash_of_compose_file>"
    }
  ]
}
```

The following table explains the attributes of the `boot-config` response:

Attribute	Description	(M)andatory or (O)ptional
pollingPeriodSecs	After successful first response of <code>boot-config</code> , the next try will be after this much delay.	O (defaults to 60secs)
files	JSON array of file JSON objects	M

Attribute	Description	(M)andatory or (O)ptional
Attributes of a file object in files JSON array		
fileType	Identifies the file type. It can be CONTAINER_FILE or COMPOSE_FILE	M
hash	SHA 256 hash of the file	M
url	Comma separated FQDN URL(s) from where the <ul style="list-style-type: none"> signed container downloadable image URL(when fileType=CONTAINER_FILE) signed docker-compose file can be downloaded (when fileType=COMPOSE_FILE) 	M (A single URL or multiple comma separated URL in case of mirrored repositories.)
container	JSON object that has attributes when filetype=CONTAINER_FILE	
container > name	Name of the container that is run based on its respective image.	M
container > imageRepoName	Container's image repo name (e.g. cli-collector)	M
container > imageRepoTag	Container's image repo tag (e.g. latest)	M

Crosswork Data Gateway's Image Manager constantly polls the Controller for the latest version of the boot-config. It looks for changes in following three attributes to determine if an image should be upgraded:

- URL of the image location
- Hash key of the image
- Image repository tag

The Controller indicates the polling interval in the boot-config. After the first poll, the Image Manager adjusts its polling interval accordingly.

Whenever Image Manager polls and finds changes in the functional images version, it downloads the new set of images, stop the old functional images container, and brings up the new functional images containers.



Note boot-config mentions the downloadable functional images. The service containers specified in the docker-compose file should be from those functional images i.e. they should match. For example, if three functional images are specified in boot-config.json, then the service containers specified in the docker-compose file should match those three functional images.

docker-compose

The docker-compose YAML file is required to bring up the Docker containers. It tells Docker which containers to launch when the VM is powered on.

The downloadable docker-compose.yml is part of `boot-config.json` provided in the Controller. The Image Manager downloads and starts the container images based on the provided docker-compose.yml.

Crosswork Data Gateway Environment Variables

Crosswork Data Gateway provides the following environment variables that can be set for each service in the docker-compose file:

Environment Variable	Use
\$CDG_ID	The Unique UUID of the Crosswork Data Gateway
\$DG_CONTROLLER_URL	URL to reach the Controller
\$DG_MGMT_IP	Self IP address on management network
\$DG_DATA_IP	Self IP address on data network
\$HTTP_PROXY	The URL of the management network proxy server
\$HTTPS_PROXY	
\$NO_PROXY	Optional comma-separated list of subnets and domains that will not be sent to the proxy server
\$DG_FTP_PROXY	
\$DG_LOG_VOLUME	Path used to mount logs from containers to the host VM

Also, you can define additional environment variables in the 'environment' section of the docker-compose file as shown in the following example:

```
environment;
  ENV_ONE: variable1
  ENV_TWO: variable2
  INSTANCE_UNIQUE_ID: ${CDG_ID}
  CONTROLLER_URL: ${DG_CONTROLLER_URL}
```



Note Initial memory is allocated to collector containers based on docker-compose configured input.

Volumes

Filesystem paths must also be specified in the Docker Compose file so that they can be mounted onto the Crosswork Data Gateway VM. Crosswork Data Gateway provides the following environment variable to point to the Crosswork Data Gateway VM root:

- \$DG_VOLUME

Following are the three supported locations for mounts:

Volume Location	Use
\$DG_LOG_VOLUME	Where container logs are mounted on the VM
\$DG_LOG_VOLUME/vitals	Where the vitals are mounted on the VM
/proc	



Note Volumes can also be defined using short syntax:

<HOST PATH>:<CONTAINER PATH>

where <HOST PATH> is one of the volume location from the table above.

For example,

```
volumes:
- /proc:/container/path/to/proc
- ${DG_LOG_VOLUME}:/container/path/to/logs
- ${DG_LOG_VOLUME}/vitals:/container/path/to/vitals
```

Docker-Compose Sample

```
version: "3.4"
services:

  management-service:
    container_name: app-manager
    image: controller:latest
    environment:
      ENV_ONE: variable1
      ENV_TWO: variable2
      INSTANCE_UNIQUE_ID: ${CDG_ID}
      CONTROLLER_URL: ${DG_CONTROLLER_URL}
    ports:
      - "6060:6060/udp"
      - "6061"
    volumes:
      - /proc:/container/path/to/proc
      - ${DG_LOG_VOLUME}:/container/path/to/logs
      - ${DG_LOG_VOLUME}/vitals:/container/path/to/vitals

  database-service:
    container_name: mysql-db
    image: MySQL:latest
    ports:
      - "5432"
    environment:
      ENV_ONE: variable1
      SAMPLE_ENV: variable2
      GATEWAY_UUID: ${CDG_ID}
    volumes:
      - /proc:/container/path/to/proc
      - ${DG_LOG_VOLUME}:/container/path/to/logs

  web-service:
    container_name: http-server
```



```

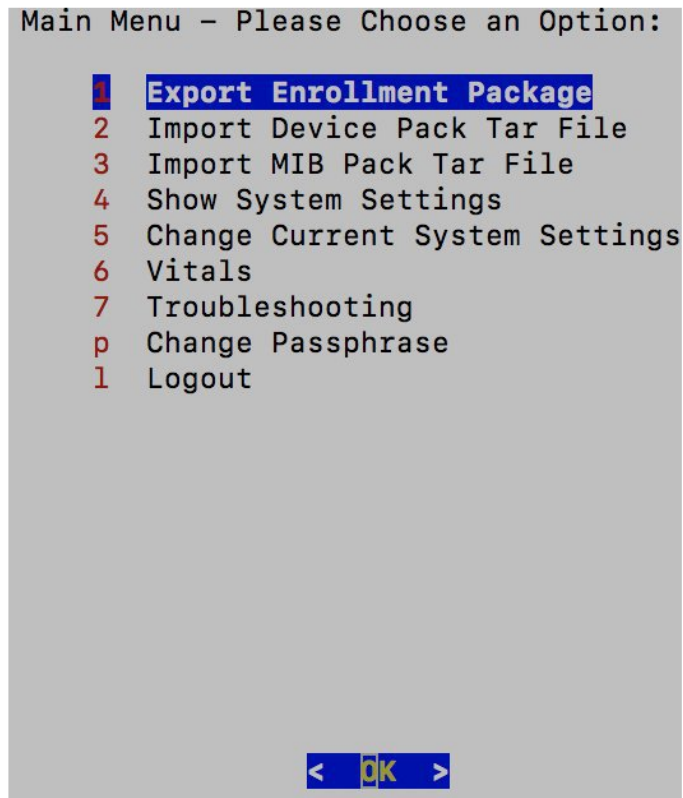
AgEGMA8GAIUdEwEB/wQF\nMAMBAf8wewYDVR0fBHQwcjA4oDagNTYyaHR0cDovL2Nybc5jb21vZG9jYs5jb20v\nQUFBQ2VydGlmawNhdGVtZXJ2aWNLcy
5jcmwwNgA0oDKGMGh0dHA6Ly9jcmwuy29t\nb2RvIm5ldC9BQUFDZXJ0aWZpY2F0ZVNLcnZpY2VzLmNybdANBgkqhkiG9w0BAQUF\nnAAOCQAQEACfb8AvCb
6P+k+tZ7xkSAzk/ExfYAWMytrwUSWgEdujm713sAg9g1o1Q\nGE&ntIghj5rCl7r+8dFRBv/38ErjHT1r0iWAFf2C3Bürz9vHCv8S5dIa2LX1rzNLz\nRt
0vxuBqw8M0Ayx91t1lawg6nQpnBBYurDC/zXDrPbDdVCYfeU0BsWO/8tqt1bgT2\nG9w84FoVxp7Z8V1IMCF1A2zs6SFz7JsDoeA3raAVGI/6ugLOpyypEB
Ms1OUIJqsi\nl2D4kF501KKaU73yqWjgom7C12yxow+ev+to51byrvLjKzg6CYGla4XXvi3tPxq3\nnsmPi9WIsqtRqAEFQ8TmDn5XpNpaY
    bg==\n-----END CERTIFICATE-----\n",
    "dg_mac_address": "00:0a:95:9d:68:16",
    "dg_ip_address": "172.23.85.145",
    "dg_user_friendly_name": "CDG1.customer.com",
    "dg_deploy_time": 1556910613,
    "dg_version": "1.0",
    "dg_resource_profile": {
    "profile_type": "Large"
    "profile": {
        "cpu": 8,
        "memory": "50",
        "nics": 3
    }
    }
}
}
}

```

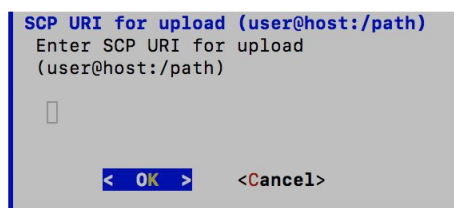
Export Enrollment Package

You can move the enrollment package from the local filesystem to an external system via SCP. Follow these steps:

Step 1 From the Main Menu, select **1 Export Enrollment Package**.



Step 2 Click **OK**. Crosswork Data Gateway prompts you to input SCP URI for uploading the enrollment package.



- Step 3** Enter the SCP URI and click **OK**.
- Step 4** Enter the SCP passphrase (the SCP user password) when prompted.
- Step 5** Click **OK**. The enrollment package is exported.
-

Static-routes Config

The Route Manager pulls the static routes configuration i.e., static-routes.json from the Controller. The static-routes config response describes the following details of each route:

- IP address of the destination network
- Next hop address
- Management and data network interface to be used

Route manager adds/deletes the static routes by comparing the existing routes configured on the VM with the contents of static-routes config pulled from the Controller.

Shown below is a sample static-routes config response:

```
{
  "routes": [
    {
      "destination_network": "10.20.0.0/16",
      "next_hop_address": "192.168.20.145",
      "nic_name": "NORTH"
    },
    {
      "destination_network": "20.1.0.0/16",
      "next_hop_address": "192.168.20.147",
      "nic_name": "SOUTH"
    },
    {
      "destination_network": "3.3.3.0/24",
      "next_hop_address": "172.23.93.23",
      "nic_name": "MGMT"
    },
    {
      "destination_network": "2001:4:4::/64",
      "next_hop_address": "2001:420:10e:2015::1",
      "nic_name": "MGMT"
    }
  ]
}
```

Crosswork Data Gateway Solutions

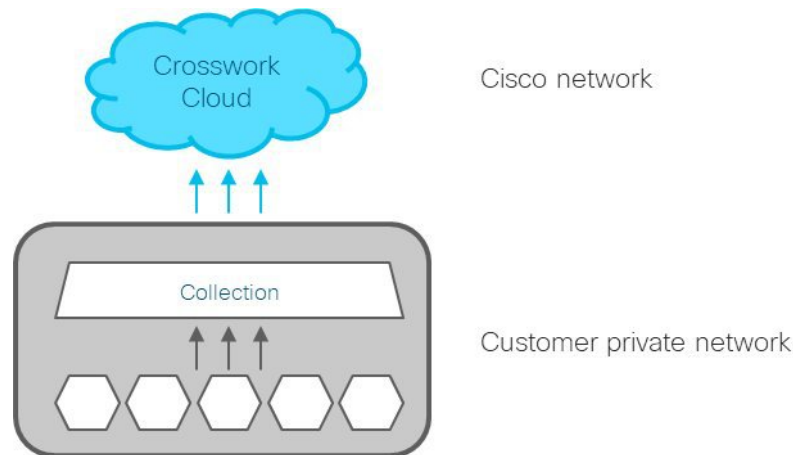
Crosswork Data Gateway has two variants:

- [Crosswork Data Gateway for Crosswork Cloud, on page 18](#)
- [Crosswork Data Gateway for Customer-hosted Solutions, on page 19](#)

Crosswork Data Gateway for Crosswork Cloud

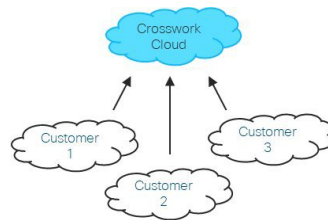
Cisco Crosswork applications face challenges collecting data from within a client's secure network. Crosswork Data Gateway provides a secure gateway that connects your private network to Cisco Crosswork Cloud. Once the connection is established, your network and Crosswork Cloud can communicate in a safe fashion to facilitate data collection.

Figure 1: Crosswork Cloud connects to the on-premise Crosswork Data Gateway for network collection



The Crosswork Cloud is designed to serve multiple customers from one Cisco hosted cloud environment. Each client has its own private network with its own on-premise collection via Crosswork Data Gateway.

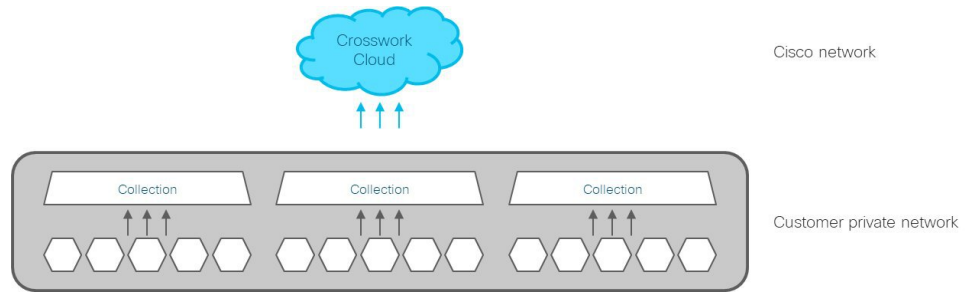
Figure 2: Crosswork Cloud serves a multi-tenant environment



Depending on your private network's size and configuration, you may require one or more Crosswork Data Gateway instances for collection. It may be necessary to deploy multiple Crosswork Data Gateway instances to address the requirements for:

1. Specific protocol clusters
2. Geo-separated regions
3. Massive scale

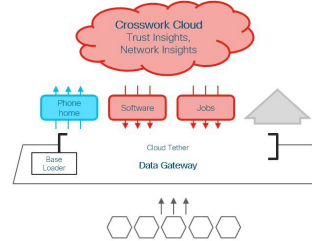
Figure 3: In mass scale networks, the collection can be distributed across multiple Crosswork Data Gateway instances



Crosswork Cloud is responsible for coordination between the multiple instances and managing them. The following figure shows the workflow of Crosswork Data Gateway in Cloud deployment scenario:

2. Data Gateway will pull all the necessary software off the parent server.

3. The Crosswork Cloud will send all the desired software and tasks to Data Gateway



1. As soon as Data Gateway is turned up it will attempt to phone home to the configured parent server.

4. Data Gateway will execute the software and perform the tasks it is instructed and collect the network data.

Legend for ownership:

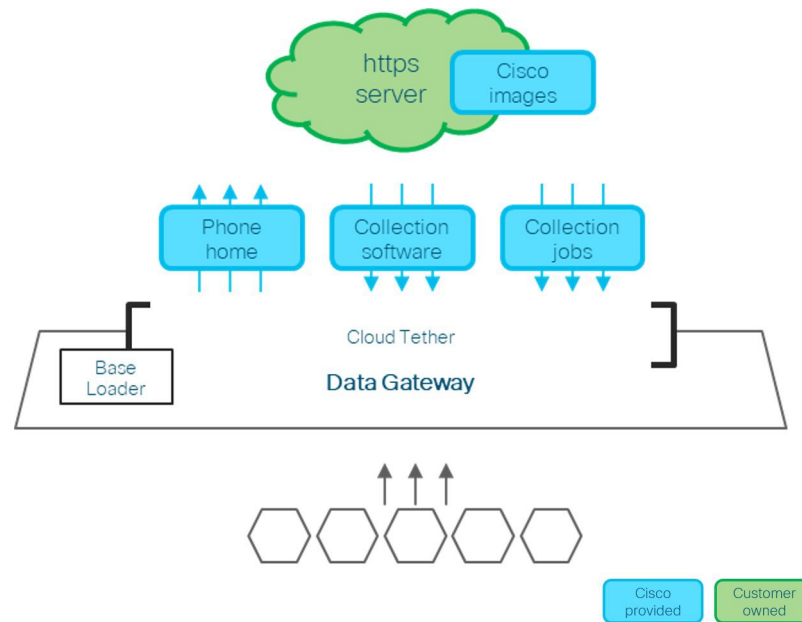
- CDG owned (blue box)
- CCC owned (red box)

Crosswork Data Gateway for Customer-hosted Solutions

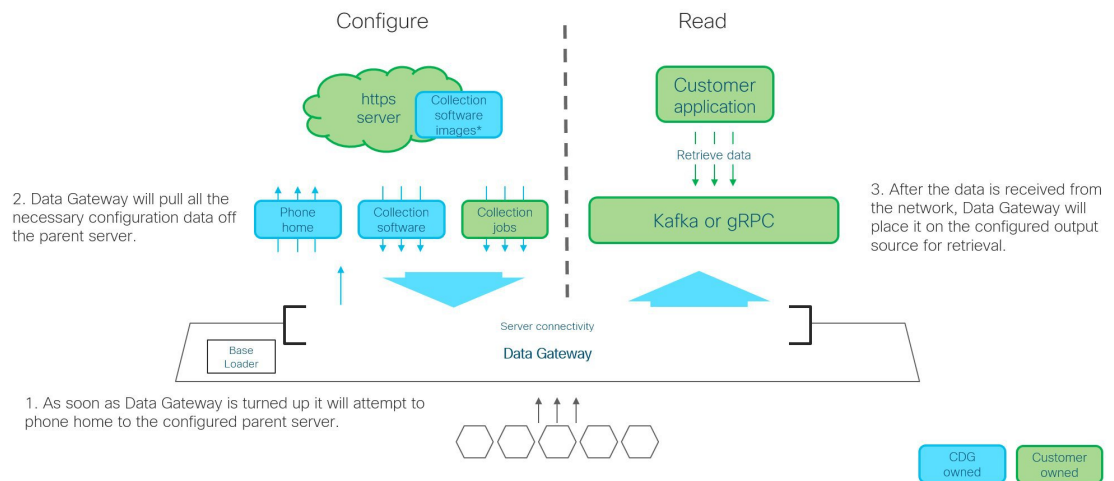
Crosswork Data Gateway for customer-hosted solutions acts as a conduit for feeding customer-hosted Kafka or gRPC server. You can host an HTTPS server within your private network that will act as the Controller in place of the Crosswork Cloud (see *Customer-hosted Solution Specific Requirements* in *Cisco Crosswork Data Gateway 1.0 Installation Guide*).

Though Cisco provides functional images to install on Crosswork Data Gateway as part of the software delivery, in this deployment scenario, you can use your custom images too.

Figure 4: Customer-hosted Server



The following figure explains the workflow of Crosswork Data Gateway in a customer-hosted solution:



Note Polled data is not requested from the device until Crosswork Data Gateway is ready to process and transmit data. This is to avoid any data loss.

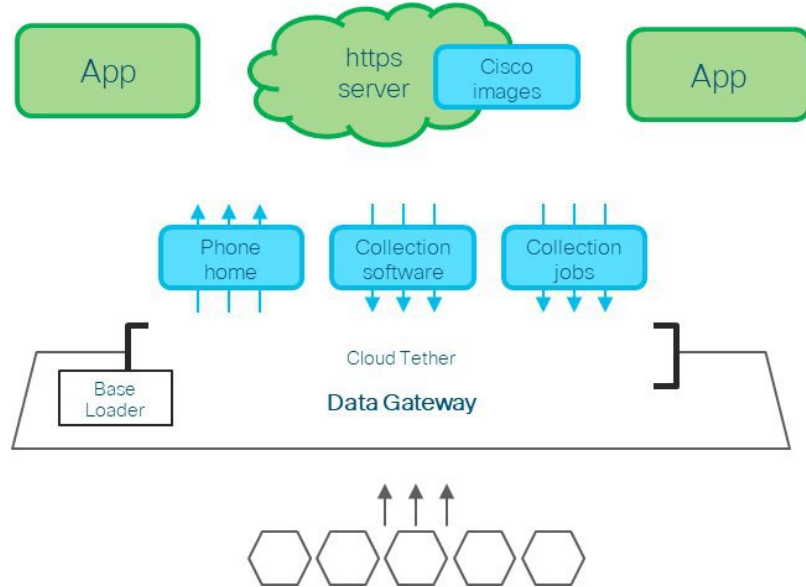
Different implementations of Crosswork Data Gateway for customer-hosted solutions

1. One Controller, Multiple Applications

You can attach multiple consuming applications for output from the same Crosswork Data Gateway instance. However, this requires additional logic on the Controller to coordinate and manage the multiple

applications. Also, in any case, there can only be one Controller managing the software and dictating the actions of a Crosswork Data Gateway instance(s).

Figure 5: One Controller, Multiple Applications



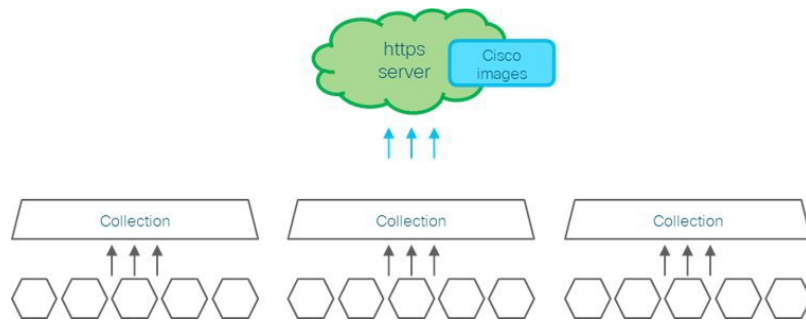
2. Multiple Crosswork Data Gateway Instances, Multiple Applications

Multiple Crosswork Data Gateway instances can be configured from the same Controller.



Note In this case, the communication and management of each instance is the responsibility of the 3rd party application and outside the scope of Crosswork Data Gateway.

Figure 6: Customer hosted application attached to multiple Data Gateways



Cisco recommends the simplest approach of a fixed configuration of devices to a particular instance (such as x to y for CDG1 and (y+1) to z for CDG2).



Note More complicated approaches for resource optimization and dynamic assignment of tasks are possible and if desired, we recommend working with Cisco Customer Experience team to design the behavior.

High-Level Workflow

The following workflow describes the main steps to getting started with Crosswork Data Gateway:

Prerequisites (applicable only to customer-hosted solutions)

The following tasks must be done before you start using Crosswork Data Gateway:

1. [Set Up Controller, on page 23](#)
2. [Set Up Output Channels, on page 23](#)

Crosswork Data Gateway Workflow

1. [Crosswork Data Gateway Authentication and Bootstrap, on page 26](#)

This includes:

1. Enroll Crosswork Data Gateway
 2. Session Establishment
 3. Download Images
2. [Add/Delete Collection Jobs, on page 53](#)
 3. View collected data in your selected output channel



CHAPTER 3

Set Up Crosswork Data Gateway

This section contains the following topics:

- [Set Up Controller, on page 23](#)
- [Set Up Output Channels, on page 23](#)
- [Crosswork Data Gateway Authentication and Bootstrap, on page 26](#)

Set Up Controller

Controller controls the collection of data and its distribution to output destinations. It also instructs the Crosswork Data Gateway where it can download functional images from.

The Controller is hosted as a separate software entity and controls the functionality of one or more Crosswork Data Gateway instances. Its implementation varies with the use case:

- In the Crosswork Cloud use case, the Controller functionality is subsumed within the Cloud application business logic.
- In the Customer-hosted solution use-case, the Controller is a custom HTTPs server provided by you, which provides a rudimentary means of managing images on Crosswork Data Gateway and specifying static collection jobs.

The Controller requirements are described in *Cisco Crosswork Data Gateway 1.0 Installation Guide*. It is important to meet these requirements for successful deployment and functioning of Crosswork Data Gateway. The Controller must be ready for integration with Crosswork Data Gateway.



Note

- Developing and integrating a Controller with Crosswork Data Gateway is out of scope of this document, however Cisco provides guidelines on how the same should be implemented.
 - A Crosswork Data Gateway instance can be integrated with only one Controller at a time.
-

Set Up Output Channels

Crosswork Data Gateway allows you to distribute collected data to either a Kafka server or a gRPC server.

To do this, Kafka or gRPC server information should be provided in the "sink" component of the collection job payload.



Note Crosswork Data Gateway is read only to the network. The devices must be configured for the correct data metrics in advance of the Crosswork Data Gateway collecting them.

Output Data Format

Crosswork Data Gateway generates output in **protobuf** format. A sample output is shown below:



Note Crosswork Data Gateway Collected Data Output **Sequence Number** resets if the Crosswork Data Gateway VM is rebooted or the collector is restarted.

Data Gateway Output protobuf message:

```

/*
 * Copyright (c) 2018 Cisco Systems, Inc.
 * All rights reserved.
 */
syntax = "proto3";

package output;
import "google/protobuf/timestamp.proto";

option java_package = "com.cisco.dg.protobuf.output";
option java_outer_classname = "DataOutput";

/*
 * Supported collector types
 */
enum CollectorType {
    INVALID_COLLECTOR_TYPE = 0;
    CLI = 1;
    SNMP = 2;
    MDT = 3;
    NETCONF = 4;
}

/*
 * CollectionSensorHeader contains all necessary fields that client app would need to
 * relate to the data that was requested, via a collection job. It will also help the
 * client app
 * to determine if the data is missing or if any cadences not met.
 */
message CollectionSensorHeader {
    string collection_job_id = 1; // collection job id via collection job request.
    string source_name = 2; // Source name via collection job request.
    string device_host = 3; // Host or ip address from which data is collected.
    string device_name = 4; // device name or tag that was given via collection job request.

    string sensor_config_id = 5; // sensor config id that identifies each sensorConfig via
    collection job request.
    CollectorType collector_type = 6; // Type(CLI/SNMP/MDT etc.) of the collector used for
    data collection.
    string sensor_path = 7; // sensor path for which the data is collected.

```

```

}

message CollectionDetailHeader {
    google.protobuf.Timestamp collection_start_time = 1;    // time when the data was
    requested from device.
    google.protobuf.Timestamp collection_end_time = 2;    // time when the data was
    received from device.
    // Incrementing number per cadence for this device and sensor path.
    // Starts at 0. Incremented before requesting data.
    int64 sequence_number = 3;
}

/*
    DataEnvelop is how the data is received by client app. Dispatcher would construct this
    message and
    stream to client app transport set in the collection job request.
*/
message DataEnvelop {
    CollectionSensorHeader sensor_header = 1; // Header to identify collection job and
    sensor path.
    CollectionDetailHeader detail_header = 2; // collection stats to identify.
    CollectionOutput output = 3; // collected data or error.
}

/*
    CollectionOutput holds Data and Error details.
*/
message CollectionOutput {
    oneof type {
        CollectionData data = 1; // Data would have value in case of success, would be null
        in case of error.
        CollectionError error = 2; // Error string in case of collection error, would be
        null in case of success.
    }
}

/*
    Data holds the collected data from the device .
*/
message CollectionData {
    // data bytes is serialized data collected in different formats by each collector. Based
    on "CollectorType" in
    // CollectionSensorHeader the serialized bytes format would vary. Following format can
    be expected per type
    //
    // CLI: UTF-8 string containing the console output text of a CLI. In case of custom XDE
    packages the format would
    // depend on output format of a custom package. (ex. UTF-8 string which contains xml
    data etc.)
    //
    // MDT: serialized Telemetry proto. (can contain GPB or GPB-KV data). XR devices expose
    this telemetry.proto.
    //
    // SNMP: serialized SnmpData proto. (For OID, Table and MIB walk operations)
    // In case of custom XDE packages the format would depend on output format of a custom
    package.
    // (ex. UTF-8 string which contains xml data etc.)

    bytes data = 1; // Data collected from device.
}

/*
    Snmp data collected for OID, Table and Mib walk operation will have the following format.
*/

```

```

message SnmpData {
    repeated OidRecord oid_records = 1;    // list of OID records.
}

message OidRecord {
    string oid = 1;                        // SNMP object identifier (OID), e.g.
    1.3.6.1.2.1.2.2.1.4.2
    string symbol = 2;                     // symbol for OID, textual / user friendly name
    representing OID, e.g. ifMtu
    string value = 3;                       // corresponding value from device, e.g. 1500
}

/*
    Error holds the details of the error occurred during processing.
*/
message CollectionError {
    string error_message = 1; // Error message to describe the reason for collection error.

    string detail_error = 2; // Detailed error message to analyse the failure.
    string error_code = 3; // Error code to determine the type of error.
}

/*
    This RPC service exposed to produce DataEnvelop.
    gRPC server has to implement this service to receive stream data.
    gRPC client has to use this service to send the stream data
*/
service OutputService {
    rpc streamData (stream DataEnvelop) returns (stream DataEnvelop) {
    };
}

```

Crosswork Data Gateway Authentication and Bootstrap

Enroll Crosswork Data Gateway

Crosswork Data Gateway generates an Enrollment package using information from the OVF template. The enrollment package is required for the unique identification of the Crosswork Data Gateway with the Controller. For more information on Enrollment Package, see [Enrollment Package, on page 14](#).

The enrollment package is uploaded to the Controller who then instantiates a new Crosswork Data Gateway object in its database and waits for a first-sign-of-life from the Crosswork Data Gateway.

Session Establishment

The Image Manager component utilizes the Base URL and CA Root Cert to setup a session with Crosswork Cloud Service. It confirms the identity of the Controller and offers its own proof of identity via signed certificates during this initial connection.

Download Images

Once the session is established, the Image Manager requests the Controller for the [boot-config, on page 10](#). The Manifest is sent as a signed JSON envelope containing a signature.

Image Manager compares the SHA ID of the images with the local cache images SHA ID and forms a list of images that needs to be upgraded/created/removed.

Image Manager downloads the signed images .tar, validates the SHA hash. It also downloads the [docker-compose, on page 12](#) file.

After that it proceeds to install and boot the containers.

Crosswork Data Gateway is now ready to collect data.



CHAPTER 4

Manage Users

This section contains the following topics:

- [Supported User Roles, on page 29](#)
- [Change Password, on page 31](#)

Supported User Roles

Cisco Crosswork Data Gateway supports only two users with the following user roles:

- **Administrator:** One default user with administrator role is created when Crosswork Data Gateway is brought up for the first time. This user cannot be deleted and has both read and write privileges such as start/shut down Crosswork Data Gateway, register an application, apply authentication certificates, configure server settings, and perform kernel upgrade.
- **Operator:** This user is also created by default during the initial VM bring up. Operator can review the state/health of the Crosswork Data Gateway, retrieve health/error logs, receive error notifications and run connectivity tests between Crosswork Data Gateway instance and the output destination.



Note

- Both users' credentials are configured during Crosswork Data Gateway installation.
- Users are locally authenticated.

The following table shows the permissions available to each role:

Table 1: Permissions Per Role

Permissions	Administrator	Operator
Export enrollment package	✓	✓
Import Device Pack Tar File	✓	×
Import MIB Pack Tar File	✓	×
Show system settings		

Permissions	Administrator	Operator
<ul style="list-style-type: none"> • Management and South/North-bound Data Addresses • NTP • DNS • Proxy • UUID • Syslog • Certificates 	✓	✓
Change Current System Settings		
<ul style="list-style-type: none"> • Configure NTP • Configure DNS • Configure Control Proxy • Configure Static Routes • Configure Syslog • Create new SSH keys • Import Certificate 	✓	✗
Vitals		
<ul style="list-style-type: none"> • Docker Containers • Docker Images • Controller Reachability • NTP Reachability • Route Table • ARP Table • Network Connections • Disk Space Usage 	✓	✓
Troubleshooting		
Ping a Host	✓	✓
Traceroute to a Host	✓	✓
NTP Status	✓	✓

Permissions	Administrator	Operator
System Uptime	✓	✓
Run show-tech	✓	✓
Remove All Collectors and Reboot VM	✓	×
Reboot VM	✓	×
Change Passphrase	✓	✓

Change Password

Both Administrator and Operator users can change their own passphrases but not each others'.

Follow these steps to change your passphrase:

Step 1 From the Main Menu, select **p Change Passphrase**.

Step 2 Click **OK**. Input your current password when prompted.

```
Changing password for dg-admin.
(current) UNIX password: [ ]
```

Step 3 Press Enter and input new password when prompted.

Step 4 Press Enter and re-type password when prompted.

```
Changing password for dg-admin.
[(current) UNIX password:
[Enter new UNIX password:
[Retype new UNIX password:
```

Step 5 Press Enter. The password is changed.



CHAPTER 5

Configure Crosswork Data Gateway

This section contains the following topics:

- [View Current System Settings, on page 33](#)
- [Change Current System Settings, on page 36](#)

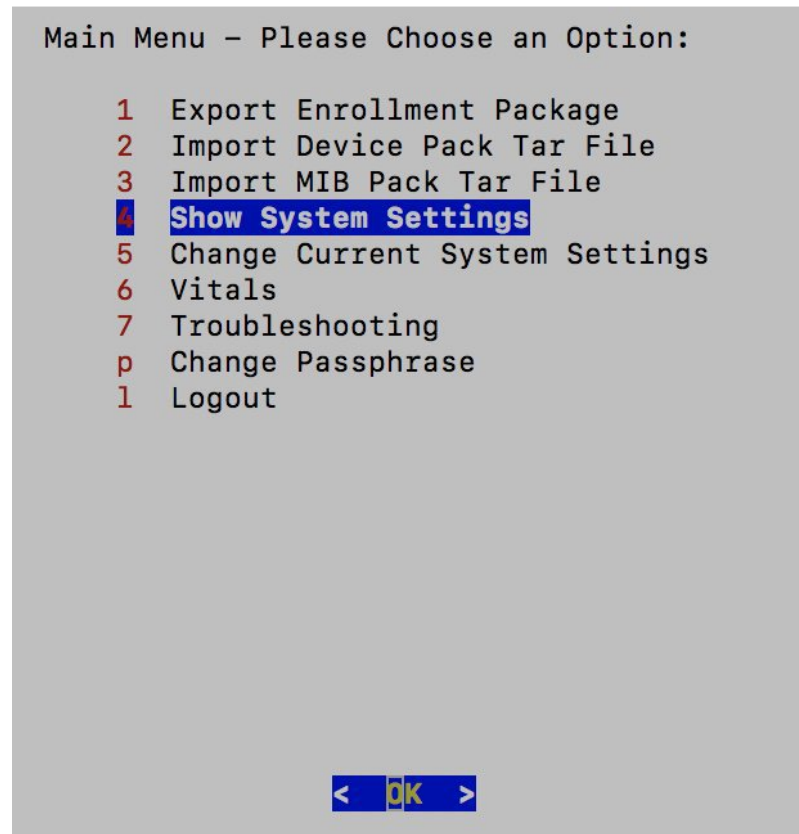
View Current System Settings

Crosswork Data Gateway allows you to view the following settings:

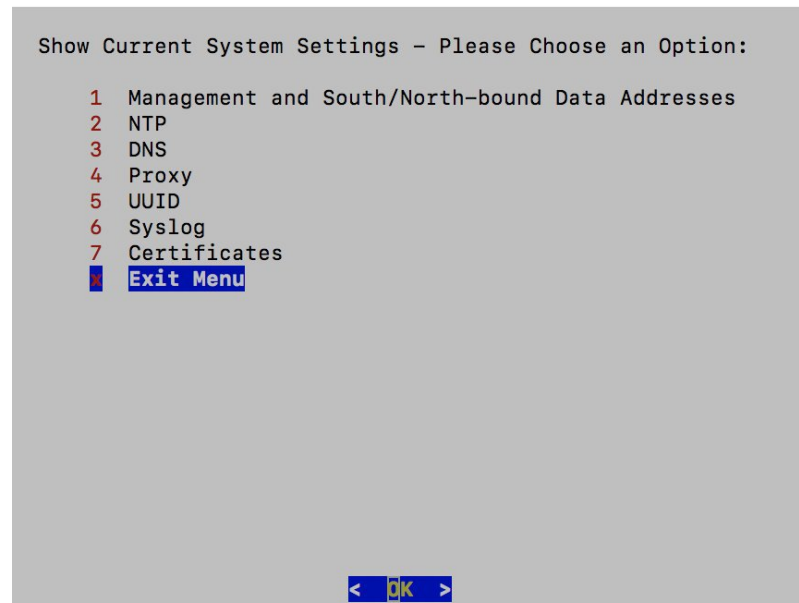
- Management and South/North-bound Data Addresses
- NTP
- DNS
- Proxy
- UUID
- Syslog
- Certificates

Follow these steps to view the current system settings:

Step 1 From the Main Menu, select **4 Show System Settings**, as shown in the following figure:



Step 2 Click **OK**. The **Show Current System Settings** menu opens.



Step 3 Select the setting you want to view.

- Step 4** Click **OK**. Crosswork Data Gateway displays the selected setting.
- After you are done viewing the settings, press any key to return to the **Show Current System Settings** menu.
- To return to the Main Menu, select **x Exit Menu** and click **OK**.
-

View Management and South/North-bound Addresses

- Step 1** From the **Show Current System Settings** Menu, select **1 Management and South/North-bound Data Addresses**.
- Step 2** Click **OK**. Crosswork Data Gateway displays the **Management and South/North-bound Data Addresses** settings.
-

View NTP Settings

- Step 1** From the **Show Current System Settings** Menu, select **2 NTP**.
- Step 2** Click **OK**. Crosswork Data Gateway displays the **NTP** settings.
-

View DNS Settings

- Step 1** From the **Show Current System Settings** Menu, select **3 DNS**.
- Step 2** Click **OK**. Crosswork Data Gateway displays the **DNS** settings.
-

View Proxy Settings

- Step 1** From the **Show Current System Settings** Menu, select **4 Proxy**.
- Step 2** Click **OK**. Crosswork Data Gateway displays the **Proxy** settings.
-

View VM UUID

- Step 1** From the **Show Current System Settings** Menu, select **5 UUID**.
- Step 2** Click **OK**. Crosswork Data Gateway displays the **UUID** of the VM.
-

View Syslogs

-
- Step 1** From the **Show Current System Settings** Menu, select **6 Syslog**.
- Step 2** Click **OK**. Crosswork Data Gateway displays the **Syslog** of the VM.
-

View Certificates

-
- Step 1** From the **Show Current System Settings** Menu, select **7 Certificates**.
- Step 2** Click **OK**. **Show Certificates** menu opens:

```
Show Certificates - Please Choose an Option:
 1 Data Gateway Signing Certificate File
 2 Controller Signing Certificate File
 3 Controller SSL/TLS Certificate File
 4 Syslog Certificate File
  Exit Menu

< OK >
```

- Step 3** Select the certification you want to view and click **OK**. Crosswork Data Gateway displays the selected certificate.
-

Change Current System Settings



Note System settings can only be configured by the Administrator.

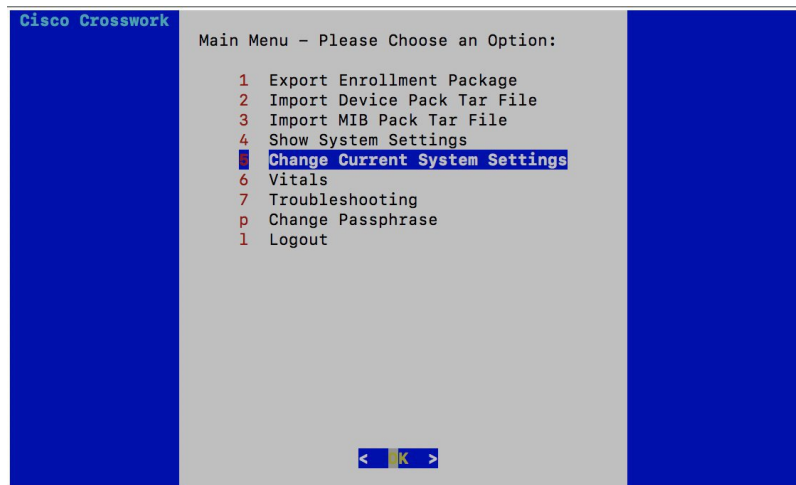
Crosswork Data Gateway allows you to change the following settings:

- NTP
- DNS

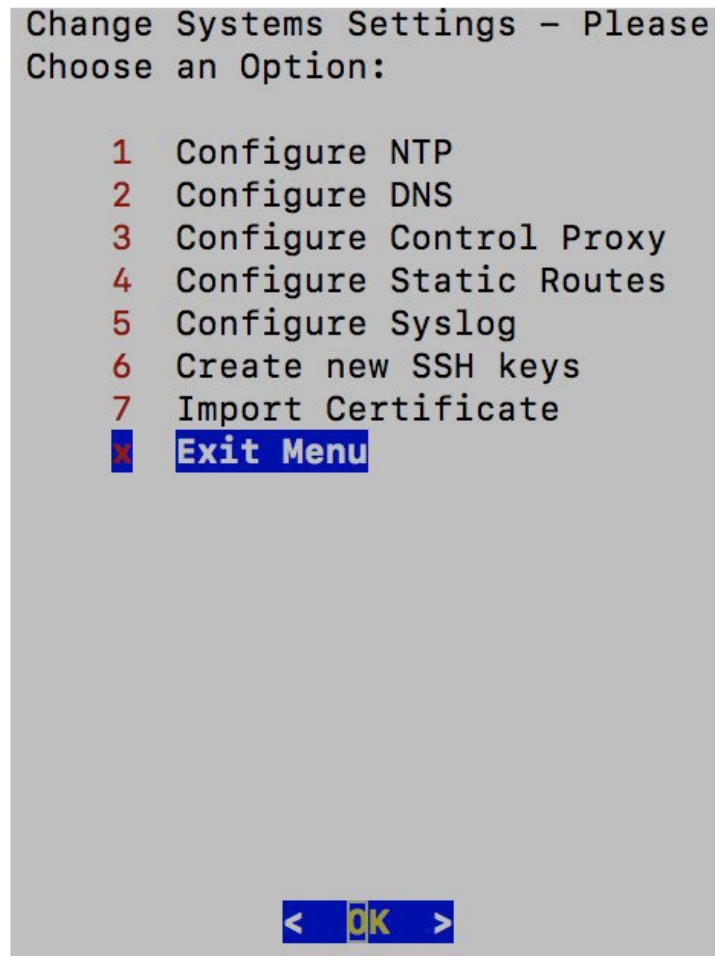
- Control Proxy
- Static routes
- Syslog
- SSH keys
- Certificate

Follow these steps to change the current system settings:

Step 1 From the Main Menu, select **5 Change Current System Settings**, as shown in the following figure.



Step 2 Click **OK**. The **Change System Settings** menu opens.



Step 3 Select the setting you want to change.

Step 4 Click **OK**. Crosswork Data Gateway prompts you to input new value for the selected setting.

Step 5 After you have entered the new settings, click **OK** to save the settings and return to the **Change System System Settings** menu.

To return to the Main Menu, select **x Exit Menu** and click **OK**.

Configure NTP

Step 1 From the **Change Current System Settings** Menu, select **1 Configure NTP**.

Step 2 Click **OK**. Enter the new NTP server address when prompted.

Step 3 Click **OK** to save the settings.

Configure DNS

- Step 1** From the **Change Current System Settings** Menu, select **2 Configure DNS**.
 - Step 2** Click **OK**. Enter the new DNS server address when prompted.
 - Step 3** Click **OK** to save the settings.
-

Configure Control Proxy

- Step 1** From the **Change Current System Settings** Menu, select **3 Configure Control Proxy**.
 - Step 2** Click **OK**. Enter the new Proxy server URL when prompted.
 - Step 3** Click **OK** to save the settings.
-

Configure Static Routes

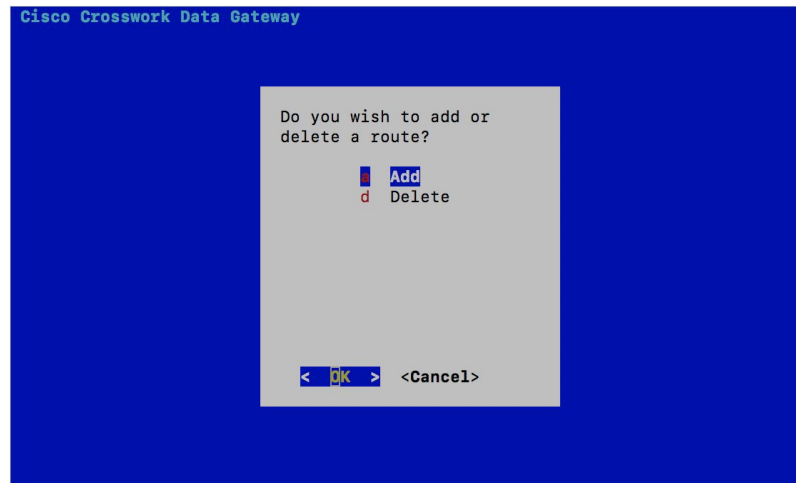
In Crosswork Data Gateway, the preferred approach to configure static routes is to configure them on the Controller. Crosswork Data Gateway's Route Manager periodically syncs the configured static routes from the Controller. Configuring static routes from the interactive menu is available only to fix any issues with static routes configuration synced from the Controller.



Note If static routes are configured directly from the interactive menu without updating static routes in Controller then those static routes will be lost next time the Route Manager syncs with the Controller. On rebooting Crosswork Data Gateway, static routes will be lost temporarily till synced again from the Controller by the Route Manager.

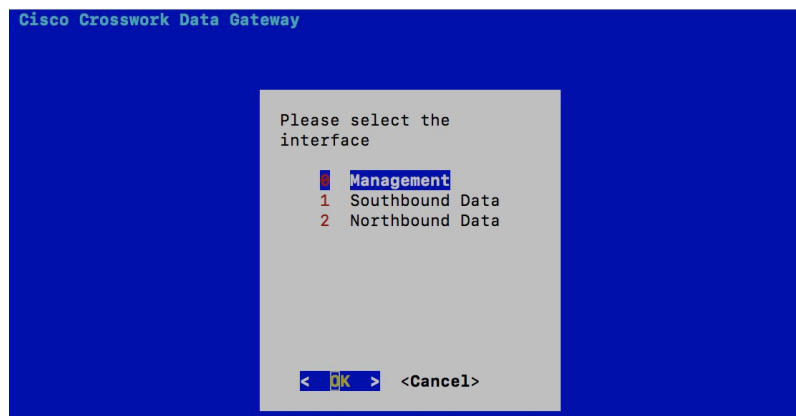
Add Static Routes

- Step 1** From the **Change Current System Settings** Menu, select **4 Configure Static Routes**.
- Step 2** Click **OK**. Crosswork Data Gateway displays the following prompt:



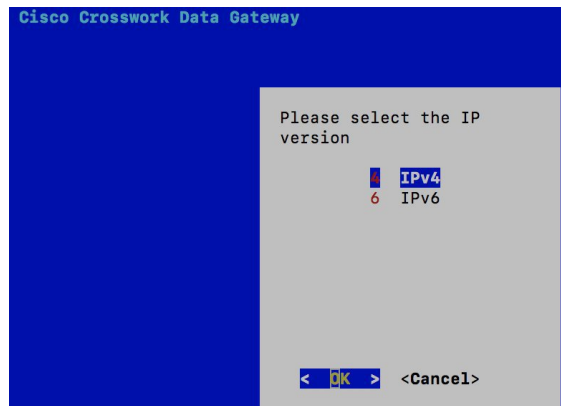
Step 3 To add a static route, select a **Add**.

Step 4 Click **OK**. Crosswork Data Gateway prompts you to select the interface:



Step 5 Select the interface for which you want to add a static route.

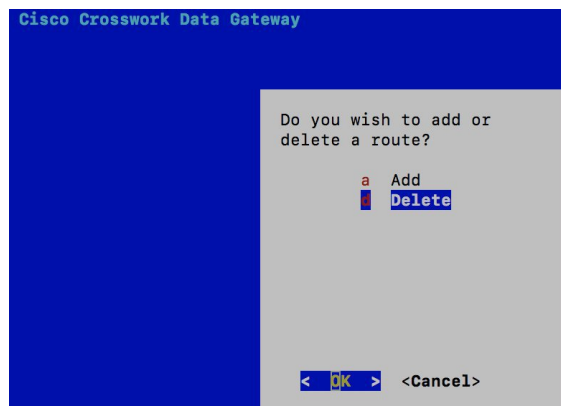
Step 6 Click **OK**. Crosswork Data Gateway prompts you to select IP address version.



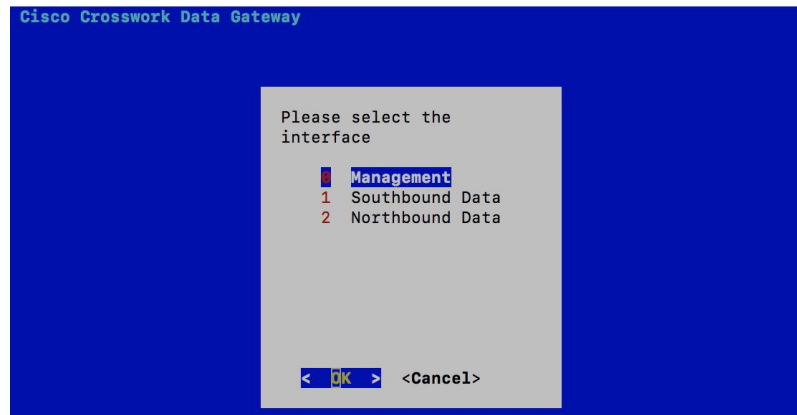
- Step 7** Select the IP address version you prefer.
- Step 8** Click **OK**. Enter IPv4/IPv6 subnet in CIDR format when prompted.
- Step 9** Click **OK**. The static route for the entered IP address is added.
-

Delete Static Routes

- Step 1** From the **Change Current System Settings** Menu, select **4 Configure Static Routes**.
- Step 2** Click **OK**. Crosswork Data Gateway displays the following prompt:

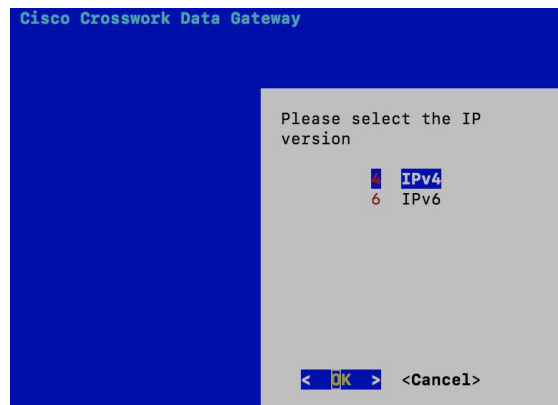


- Step 3** To delete a static route, select **d Delete**.
- Step 4** Click **OK**. Crosswork Data Gateway prompts you to select the interface:



Step 5 Select the interface for which you want to delete a static route.

Step 6 Click **OK**. Crosswork Data Gateway prompts you to select IP address version.



Step 7 Select the IP address version you prefer.

Step 8 Click **OK**. Enter IPv4/IPv6 subnet in CIDR format when prompted.

Step 9 Click **OK**. The static route for the entered IP address is delete.

Configure Syslog



Note For any Syslog server configuration with IPv4/IPv6 support for different linux distributions, please refer your system administrator and configuration guides.

Step 1 From the **Change Current System Settings** Menu, select **5 Configure Syslog**.

Step 2 Click **OK**. Enter the new Syslog server address or hostname when prompted.

Step 3 Click **OK** to save the settings.

Create New SSH Keys

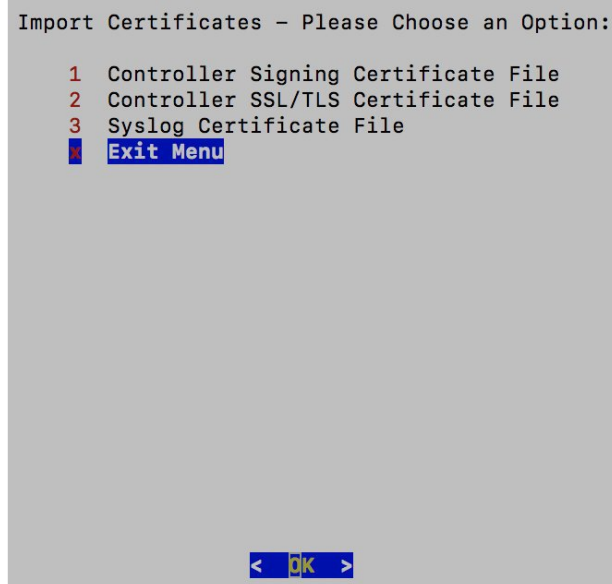
Step 1 From the **Change Current System Settings** Menu, select **6 Create new SSH keys**.

Step 2 Click **OK**. Crosswork Data Gateway launches an auto-configuration process that generates new SSH keys.

Import Certificate

Step 1 From the **Change Current System Settings** Menu, select **7 Import Certificate**.

Step 2 Click **OK**. Select the certificate to be imported when prompted.



Step 3 Select the certificate you want to import.

Step 4 Click **OK** and enter SCP URI to the selected certificate file when prompted.

Step 5 Click **OK** and enter passphrase for the SCP URI.



CHAPTER 6

Collection Jobs

This section contains the following topics:

- [About Collection Jobs, on page 45](#)
- [Collection Job Payload Model, on page 46](#)
- [Add/Delete Collection Jobs, on page 53](#)
- [Best Practices and Limitations for Creating Collection Jobs, on page 54](#)
- [Sample Collection Jobs Payloads, on page 55](#)

About Collection Jobs

Crosswork Data Gateway pulls functional images from the Controller. Each functional image represents a collection profile. These collection profiles describe what task the Controller is expected to perform.

You can collect more than one type of data at a time by using separate collection profiles.

Each collection job you create executes the collection requests and receives the collection data in their respective protocol/transport. The collected raw data from the devices and then fed to either Kafka or gRPC server.



Note Crosswork Data Gateway is read only to the network. The devices must be configured for the correct data metrics in advance of the Crosswork Data Gateway collecting them.

Cisco Crosswork Data Gateway lets you create three types of collection jobs:

CLI Collection Job

Enables CLI-based data collection (such as device configuration) from the network devices. The CLI collector uses XDE/PAL to collect device data for a given CLI. Only **show** commands are supported for this type of collection job.

SNMP Collection Job

Enables SNMP-based data collection based on the OIDs supported on the devices.

Supported SNMP versions include SNMPv1, SNMPv2c, and SNMPv3 for data polling and traps.

MDT Collection Job

Collects model driven telemetry data streamed from the device to the Crosswork Data Gateway.

Collection Job Payload Model

A collection job describes the following:

- Data to be collected
- Devices from which collection is desired and credentials to authenticate
- Collection intervals (a periodic interval no greater than 30 days or immediately on demand)

A single collection job can contain either CLI commands, SNMP MIB requests, or MDT subscriptions. Crosswork Data Gateway routes the request to the appropriate collector to fetch the requested data.

A collection job has three main parts:

```
//      Device Groups(s) -- identifies the different device groups from which data is
//                          to be collected and their authentication credentials.
//      Source(s)      -- identifies the different sensors and their collection details
//                          like sensor path , devices, etc
//      Sink(s)        -- identifies the different output destination the final CDG data
//
//                          is sent to
//
//
```

A typical collection request contains at least the following information:

- Collection Job ID
- Method of collection
- Device Groups
- Sensor Path (CLI, SNMP, or MDT)
- Cadence or Collection Interval
- Output Target

Shown below is a sample collection job payload:

```
{
  "collection-job-id": "<sample-collection-job>",
  "collectorType": "CLI",
  "deviceGroups": [
    {
      "id": "deviceGroup1",
      "devices": [
        {
          "name": "vm1-xrv9000",
          "host": "<device_IP_address>"
        },
        {
          "name": "vm2-xrv9000",
          "host": "<device_IP_address>"
        }
      ]
    }
  ],
}
```

```

    {
      "name": "vm3-xrv9000",
      "host": "<device_IP_address>"
    }
  ],
  "deviceParams": [
    {
      "key": "software",
      "value": "IOS"
    },
    {
      "key": "transport",
      "value": "SSH2"
    },
    {
      "key": "port",
      "value": "22"
    },
    {
      "key": "username",
      "value": "admin"
    },
    {
      "key": "password",
      "value": "<password>"
    },
    {
      "key": "vendor",
      "value": "cisco"
    }
  ]
},
{
  "id": "deviceGroup2",
  "devices": [
    {
      "name": "vm1-xrv9000",
      "host": "<device_IP_address>"
    },
    {
      "name": "vm2-xrv9000",
      "host": "<device_IP_address>"
    }
  ],
  "deviceParams": [
    {
      "key": "software",
      "value": "IOS"
    },
    {
      "key": "transport",
      "value": "SSH2"
    },
    {
      "key": "port",
      "value": "22"
    },
    {
      "key": "username",
      "value": "admin"
    },
    {
      "key": "password",
      "value": "<password>"
    }
  ]
}

```

```

    },
    {
      "key": "vendor",
      "value": "cisco"
    }
  ]
}
],
"sources": [
  {
    "name": "source1",
    "deviceGroupIds": [
      "deviceGroup1",
      "deviceGroup2"
    ],
    "sensorConfigs": [
      {
        "sensorConfigId": "sensor-id-1",
        "path": "show interfaces",
        "cadence": 90,
      }
    ]
  }
],
"sinks": [
  {
    "name": "sink2",
    "parentIds": [
      "source1"
    ],
    "outputTarget": {
      "messageBusTarget": {
        "brokerAddresses": [
          {
            "host": "kafka-operational",
            "port": 9092
          }
        ],
        "username": "kafka-admin",
        "password": "password",
        "destinationTopic": "externalOutputTopic",
        "certificateKey": "BasicOne"
      }
    }
  }
],
{
  "name": "sink2",
  "parentIds": [
    "source"
  ],
  "outputTarget": {
    "grpcTarget": {
      "password": "<password>",
      "username": "admin",
      "certificateKey": "some certificate",
      "grpcAddress": {
        "port": 60061,
        "host": "<gRPC_server_IP_address>"
      }
    }
  }
}
}

```

```

],
  "secretKey": "skey"

```

The following table explains the fields of the above payload:

Field	Type	(M)andatory or (O)ptional	Description
collection-job-id	string	M	String for unique identification of the collection job.
collectorType	string	M	Acceptable values are MDT, CLI or SNMP to indicate the type of job.
deviceGroups	json array	M	An array of information on device details. Each device group contains the details of one device type.
deviceGroups[x] > id	string	M	String for the unique identification of the device group.
deviceGroups[x] > devices	json array	M	An array of device names and IP addresses.
deviceGroups[x] > devices[y] > name	string	M	Name of the device.
deviceGroups[x] > devices[y] > host	string	M	IP address of the device.
deviceGroups[x] > devicesParams	json array	M	An array of information on devices' authentication credentials.
deviceGroups[x] > devicesParams[y] > key	string	M	Key required for device authentication.
deviceGroups[x] > devicesParams[y] > value	string	M	Value for the key to authenticate the device.
sources	json array	M	An array of information on source details that can have multiple source info(JSON objects).
sources[x] > name	string	M	A name for the source.
sources[x] > deviceGroupIds	json array	M	An array containing names of the device groups from which data has to be collected.

Field	Type	(M)andatory or (O)ptional	Description
sources[x] > sensorConfigs	json array	M	<p>An array of Sensor Configs (JSON Objects). Each Sensor Config (JSON object) element contains config details of one collection.</p> <p>At least one sensor config per collection job is a must.</p> <p>Note For a registered application, each source of sensor config should have a unique string name.</p>
sources[x] > sensorConfigs[y] > sensorConfigId	string	M	Unique ID of the sensor config.
sources[x] > sensorConfigs[y] > path	string	M	Mandatory path of sensor config containing the CLI command in case of CLI collector OR SNMP OID in case of SNMP collector OR YANG PATH in case of MDT collection. In case of MDT collection, path may also contain *.* indicating to send all sensors output for this collection job.
sources[x] > sensorConfigs[y] > cadence	int	O	Optional cadence value of this sensor config in seconds. If not passed the sensor config collection is done once.
sinks	json array	M	<p>This contains an array of destination sink information (JSON objects).</p> <p>This can provide details on one or more sinks of same/different types. Currently supported sink types are Kafka and gRPC.</p>
sink[x] > name	string	M	A unique name for the sink.

Field	Type	(M)andatory or (O)ptional	Description
sink[x] > parentIds	string array	M	An array of parent IDs from which the sink should get its input.
sink[x] > outputTarget	json object	M	A JSON object that holds the final transport information. It can hold one or more type of target information. Currently supported target types are kafka and gRPC
secretKey	string	O	A key that is needed for validating the collection job.
Sink with Kafka output target			
sinks[x] > outputTarget > messageBusTarget	json object	M	This contains connection information of the target Kafka server.
sinks[x] > outputTarget > messageBusTarget > brokerAddresses	json object	M	Array containing host+port information.
sinks[x] > outputTarget > messageBusTarget > brokerAddresses > host	string	M	An IP address or reachable name of the Kafka server.
sinks[x] > outputTarget > messageBusTarget > brokerAddresses > port	int	M	Port number based on transport protocol to be used to connect to the Kafka server.
sinks[x] > outputTarget > messageBusTarget > username	string	M	Credential username to connect to the Kafka server. Note Username and password are not supported in Crosswork Data Gateway 1.0. Any value passed is ignored.

Field	Type	(M)andatory or (O)ptional	Description
sinks[x] > outputTarget > messageBusTarget > password	string	M	Credential password to connect to the Kafka server. Note Username and password are not supported in Crosswork Data Gateway 1.0. Any value passed is ignored.
sinks[x] > outputTarget > messageBusTarget > destinationTopic	string	M	Name of the destination Kafka topic.
sinks[x] > outputTarget > messageBusTarget > certificateKey	bytes	M	Certificate Blob to connect to the Kafka server.
Sink with gRPC output target			
sink[x] > outputTarget > grpctarget	json object	M	This contains connection information of the target gRPC.
sink[x] > outputTarget > grpctarget > grpcaddress	json object	M	Array containing host+port information.
sink[x] > outputTarget > grpctarget > grpcaddress > host	string	M	An IP address or reachable name of the gRPC server.
sink[x] > outputTarget > grpctarget > grpcaddress > port	string	M	Port number based on transport protocol to be used to connect to the gRPC server.
sink[x] > outputTarget > grpctarget > username	string	M	Credential userid to connect to the gRPC server. Note Username and password are not supported in Crosswork Data Gateway 1.0. Any value passed is ignored.

Field	Type	(M)andatory or (O)ptional	Description
sink[x] > outputTarget > grpctarget > password	string	M	Credential password to connect to the gRPC server. Note Username and password are not supported in Crosswork Data Gateway 1.0. Any value passed is ignored.
sink[x] > outputTarget > grpctarget > certificateKey	bytes	M	Certificate Blob to connect to the gRPC server.

Add/Delete Collection Jobs

You can refer the [Sample Collection Jobs Payloads, on page 55](#) to create collection jobs. Once the collection job is created, it must be added to the collection profile i.e., functional image of the respective collector.

The collection job is picked up by the Crosswork Data Gateway for execution when the Image Manager syncs with the Controller and retrieves the latest boot-config and docker-compose.



Note

- Crosswork Data Gateway is read only to the network. The devices must be configured for the correct data metrics in advance of the Crosswork Data Gateway collecting them.
- The name of the collection job json file must be supplied (without extension ".json") as the value of the attribute **collectionJobId** in the collection payload i.e., in the content of the file. Crosswork Data Gateway throws an error if the value of **collectionJobId** does not match the name of the file.

Upon successful creation of collection job, if the output server is up, running, and accessible, Crosswork Data Gateway starts sending data to it. In this scenario, status for per device per sensor config is shown as ACTIVE in the collector logs.

However, if output server is inaccessible, Crosswork Data Gateway fails to send data to it.

On receiving error while sending data to external sink (gRPC or Kafka):

1. Sink is set to ERROR state and all the collection job(s) related to that sink are stopped for further collection.
2. A notification is sent that Sink is in ERROR state.
3. Reconnect sink flow periodically sends the last collection data to the sink until the sink resumes or the collection job(s) are deleted.
4. During this time, a warn message is logged for any inflight collection data (or) data received from device for Traps or MDT.
5. Once sink resumes, a notification is sent to inform that Sink is ACTIVE.

6. All the collection job(s) of the sink that were stopped are resumed and data is dispatched.

To delete a collection job, simply remove it from the collection profile.

Upon job deletion, the collector will send one alert to the Controller for that job once all the devices per sensor path execution has stopped.

Best Practices and Limitations for Creating Collection Jobs

Cisco recommends that following best practices be followed while creating collection job payloads:

Limitation	Best Practices
Scale	
<p>Maximum size of sensor path collected data is 10 MB. Crosswork Data Gateway collector data at the same time from N devices cannot exceed 6 GB.</p>	<ol style="list-style-type: none"> Any collected sensor path's data size should not be > 10 MB when target server is Kafka, otherwise you will get the following error message: RecordTooLargeException: The message is xxxxxxxx bytes when serialized which is larger than the maximum request size you have configured with the max.request.size configuration. If external target server is Kafka, then the following should be configured in Kafka's server properties: <ul style="list-style-type: none"> message.max.bytes=10485760 replica.fetch.max.bytes=10485760 max.request.size=10485760 fetch.message.max.bytes=10485760 Crosswork Data Gateway collector data at the same time from N devices with M sensor path per device and with P average size of collected data per sensor path cannot exceed 6 GB i.e., (N = # of devices) x (M = # of sensor path) x (P= Average Message size per sensor path) < 6GB
Log Purge Policy	
<p>When total log file size for a collector reaches 2 GB, Crosswork Data Gateway starts cleaning up by removing the old log files.</p>	<p>If you would like to save the old log files, save it to any other remote server before the total log file size reaches 2 GB per collector. This can be done by running show-tech from Crosswork Data Gateway Main Menu > Troubleshooting.</p>

Sample Collection Jobs Payloads

This section contains sample collection job payloads for the following collection profiles:

- [CLI Collection Job, on page 55](#)
- [SNMP Collection Jobs, on page 65](#)
- [MDT Collection Job, on page 84](#)

CLI Collection Job

Crosswork Data Gateway supports CLI-based data collection from the network devices. It uses XDE/PAL to collect device data for a given CLI. Only show commands are supported for this type of collection job.



Note

- While creating a CLI collection job, you must validate it for any errors. Crosswork Data Gateway does not validate collection jobs. Hence, even though a collection job is invalid, its status is still shown as **Active**.
- Device should not have any banner configuration for CLI collection to work properly. Please refer to device documentation on how to turn this off.
- The value of **Cadence** is in seconds. It should be set either to 0 to indicate the sensor configured to be collected only once.
OR
It should be ≥ 60 (i.e. at least 1 minute) up to 2764800 seconds (i.e. at most 32 days) max, indicating how frequently configured sensor data should be collected.
- When collection from a device is skipped due to previous execution still in progress, Crosswork Data Gateway raises a warning log. No alert is generated for this scenario.

Following are some CLI collection job examples:

API URL	{contextPath}/v1/collect-jobs/{jobId} Note This API resides on the Controller.
HTTP Method	GET
Response Status Code	200 OK

CLI sample payload with Kafka output target	
---	--

```
{
  "collection-job-id":
  "<unique_collection_job_name>",
  "collectorType": "CLI",
  "deviceGroups": [
    {
      "id": "deviceGroup1",
      "devices": [
        {
          "name": "vm1-xrv9000",
          "host": "<device_IP_address>"
        },
        {
          "name": "vm2-xrv9000",
          "host": "<device_IP_address>"
        },
        {
          "name": "vm3-xrv9000",
          "host": "<device_IP_address>"
        }
      ],
      "deviceParams": [
        {
          "key": "software",
          "value": "IOS"
        },
        {
          "key": "transport",
          "value": "SSH2"
        },
        {
          "key": "port",
          "value": "22"
        },
        {
          "key": "username",
          "value": "admin"
        },
        {
          "key": "password",
          "value": "<password>"
        },
        {
          "key": "vendor",
          "value": "cisco"
        }
      ]
    },
    {
      "id": "deviceGroup2",
      "devices": [
        {
          "name": "vm1-xrv9000",
          "host": "<device_IP_address>"
        },
        {
          "name": "vm2-xrv9000",
          "host": "<device_IP_address>"
        }
      ]
    }
  ]
}
```

```

    ],
    "deviceParams": [
      {
        "key": "software",
        "value": "IOS"
      },
      {
        "key": "transport",
        "value": "SSH2"
      },
      {
        "key": "port",
        "value": "22"
      },
      {
        "key": "username",
        "value": "admin"
      },
      {
        "key": "password",
        "value": "<password>"
      },
      {
        "key": "vendor",
        "value": "cisco"
      }
    ]
  },
  "sources": [
    {
      "name": "source1",
      "deviceGroupIds": [
        "deviceGroup1",
        "deviceGroup2"
      ],
      "sensorConfigs": [
        {
          "sensorConfigId":
"sensor-id-1",
          "path": "show interfaces",
          "cadence": 90,
        }
      ]
    }
  ],
  "sinks": [
    {
      "name": "sink2",
      "parentIds": [
        "source"
      ],
      "outputTarget": {
        "messageBusTarget": {
          "brokerAddresses": [
            {
              "host":
"kafka-operational",
              "port": 9092
            }
          ],
          "username": "kafka-admin",
          "password": "<password>",
          "destinationTopic":

```

	<pre>"externalOutputTopic", "certificateKey": "BasicOne" } }], "secretKey": "skey" }</pre>
Sink with gRPC output target	<pre>"sinks": [{ "name": "sink2", "parentIds": ["source"], "outputTarget": { "grpcTarget": { "password": "<password>", "username": "admin", "certificateKey": "some certificate", "grpcAddress": { "port": 60061, "host": "<gRPC_server_IP_address>" } } } }]</pre>

Sink with Kafka output target

Sample payload for CLI with custom device package.

Create a custom package and mention the function and package in the sensorPathParams in the sensorConfigs.

In this case, path is not considered.

Note The custom device package can be imported into the container using **Import Device Pack Tar File** option from the Main Menu of Crosswork Data Gateway. See [Deploy a Custom Device Package, on page 89](#).

```

{
  "collection-job-id":
  "<unique_collection_job_name>",
  "collectorType": "CLI",
  "deviceGroups": [
    {
      "id": "deviceGroup1",
      "devices": [
        {
          "name": "vm1-xrv9000",
          "host": "<device_IP_address>"
        },
        {
          "name": "vm2-xrv9000",
          "host": "<device_IP_address>"
        },
        {
          "name": "vm3-xrv9000",
          "host": "<device_IP_address>"
        }
      ],
      "deviceParams": [
        {
          "key": "software",
          "value": "IOS"
        },
        {
          "key": "transport",
          "value": "SSH2"
        },
        {
          "key": "port",
          "value": "22"
        },
        {
          "key": "username",
          "value": "admin"
        },
        {
          "key": "password",
          "value": "<password>"
        },
        {
          "key": "vendor",
          "value": "cisco"
        }
      ]
    }
  ],
  "sources": [
    {
      "name": "source1",
      "deviceGroupIds": [
        "deviceGroup1"
      ],
      "sensorConfigs": [
        {
          "sensorConfigId": "sensor-id-1",
          "path": "*",
          "cadence": 60,
          "sensorPathParams": [
            {
              "key": "function",
              "value": "show_running_config"
            }
          ]
        }
      ]
    }
  ]
}

```

```
    },
    {
      "key": "package",
      "value": "show_running_config"
    }
  ]
}
],
"sinks": [
  {
    "name": "sink2",
    "parentIds": [
      "source"
    ],
    "outputTarget": {
      "messageBusTarget": {
        "brokerAddresses": [
          {
            "host": "kafka-operational",
            "port": 9092
          }
        ],
        "username": "kafka-admin",
        "password": "password",
        "destinationTopic":
"externalOutputTopic",
        "certificateKey": "BasicOne"
      }
    }
  }
],
"secretKey": "skey"
}
```

gRPC Output Target

Sample payload for CLI with custom device package.

```

{
  "collection-job-id":
  "<unique_collection_job_name>",
  "collectorType": "CLI",
  "deviceGroups": [
    {
      "id": "deviceGroup1",
      "devices": [
        {
          "name": "vm1-xrv9000",
          "host": "<device_IP_address>"
        },
        {
          "name": "vm2-xrv9000",
          "host": "<device_IP_address>"
        },
        {
          "name": "vm3-xrv9000",
          "host": "<device_IP_address>"
        }
      ],
      "deviceParams": [
        {
          "key": "software",
          "value": "IOS"
        },
        {
          "key": "transport",
          "value": "SSH2"
        },
        {
          "key": "port",
          "value": "22"
        },
        {
          "key": "username",
          "value": "admin"
        },
        {
          "key": "password",
          "value": "<password>"
        },
        {
          "key": "vendor",
          "value": "cisco"
        }
      ]
    }
  ],
  "sources": [
    {
      "name": "source1",
      "deviceGroupIds": [
        "deviceGroup1"
      ],
      "sensorConfigs": [
        {
          "sensorConfigId": "sensor-id-1",
          "path": "*",
          "cadence": 60,
          "sensorPathParams": [
            {
              "key": "function",
              "value": "show_running_config"
            }
          ]
        }
      ]
    }
  ]
}

```

```

    },
    {
      "key": "package",
      "value": "show_running_config"
    }
  ]
}
],
"sinks": [
  {
    "name": "sink2",
    "parentIds": [
      "source"
    ],
    "outputTarget": {
      "grpcTarget": {
        "password": "<password>",
        "username": "admin",
        "certificateKey": "some
certificate",
        "grpcAddress": {
          "port": 60061,
          "host": "<gRPC_server_IP_address>"
        }
      }
    }
  }
],
"secretKey": "skey"
}

```

SNMP Collection Jobs

Crosswork Data Gateway supports SNMP-based data collection based on the OIDs supported on the devices.

The SNMP collector makes a poll request to the Controller to get its configuration profile (a list of MIB objects to collect and a list of devices to fetch from). It determines the corresponding OIDs by looking up the pre-packaged list of MIB modules or the custom list of MIB modules.

Once the OIDs are resolved, they are provided as input to the SNMP collectors.

The customization of the output can be achieved by generating a device package for the same. The device packages can be downloaded to the Crosswork Data Gateway VM using **Import MIB Pack Tar File** option from the Main Menu of Crosswork Data Gateway. See [Deploy a Custom MIB, on page 90](#).

The following SNMP versions are supported:

- SNMPv1
- SNMPv2c
- SNMPv3

The table below lists supported privacy protocols and the value that needs to be given in the collection payload for SNMP and SNMP Trap collection jobs:

Protocol	SNMP Collection Payload	SNMP Trap Collection Payload
aes	AES	N/A
des56	DES	DES
3des	3DES	3DES
aes 128	AES128	AES128
aes 192	AES192 or CiscoAES192(Cisco specific)	AES192 or CiscoAES192(Cisco specific)
aes 256	AES256 or CiscoAES256(Cisco specific)	AES256 or CiscoAES256(Cisco specific)

**Note**

- While creating an SNMP collection job, you must validate it for any errors. Crosswork Data Gateway does not validate collection jobs. Hence, even though a collection job is invalid, its status is still shown as **Active**.
- The value of **Cadence** is in seconds. It should be set either to 0 to indicate the sensor configured to be collected only once.

OR

It should be ≥ 60 (i.e. at least 1 minute) up to 2764800 seconds (i.e. at most 32 days) max, indicating how frequently configured sensor data should be collected.

- When collection from a device is skipped due to previous execution still in progress, Crosswork Data Gateway raises a warning log. No alert is generated for this scenario.
- For SNMP v1/v2c, if the device details (such as host or community string) are incorrect in the payload, Crosswork Data Gateway ignores the traps received from the device and logs the following WARN message:

```
23:38:50.566 [Trap.0] WARN c.c.d.s.t.receiver.
SnmTrapReceiver - Ignoring trap received. No collection
-job found for device: <ip>, securityName: <communitystring>,
version: <version>
```

In case of SNMP v3, if the device details (such as auth, priv, and security name details) are incorrect in the payload, Crosswork Data Gateway filters it out and hence, does not receive the trap. Thus, no WARN message is logged.

Sample Configurations on Device:
**Note**

Device configuration is customer's responsibility as Crosswork Data Gateway does not perform any device configuration activity.

Version	Configuration
V1	<pre>snmp-server group group1 v1 snmp-server user user1 group1 v1 snmp-server host <host_ip> traps <community_string> udp-port 1062</pre> <p>For example,</p> <pre>snmp-server host 172.29.194.78 traps test udp-port 1062</pre> <p>Note Version 1 is the default version used by the device.</p>
V2c	<pre>snmp-server group group1 v2c snmp-server user user1 group1 v2c snmp-server host 172.29.194.142 traps version 2c v2test udp-port 1062</pre>
V3	<pre>snmp-server group group1 v3 auth notify user1 read user1 write user1 snmp-server view user1 1.3 included snmp-server user user1 group1 v3 auth md5 <password> priv aes 128 <password> snmp-server host 172.23.92.193 traps version 3 priv user1 udp-port 1062</pre>

The SNMP Collector supports the following operations:

- SCALAR
- TABLE
- MIB_WALK
- TRAP
- DEVICE_PACKAGE

These operations are defined in the sensor config (see payload example below).



Note There is an optional **deviceParams** attribute **snmpRequestTimeoutMillis** (not shown in the sample payloads) that should be used if the device response time is very high. It's not recommended to use **snmpRequestTimeoutMillis** unless you are absolutely certain that your device response time is very high.

The value for **snmpRequestTimeoutMillis** should be specified in milliseconds:

Default value is 1500 milliseconds

Minimum value is 1500 milliseconds

However, there is no limitation on the maximum value of this attribute.

SNMP Collection Job

The following table shows different types of SNMP collection jobs:

API URL	{contextPath}/v1/collect-jobs/{jobId}
HTTP Method	GET
Response Status Code	200 OK

SNMP payload example with Kafka as output target	
---	--

```

{
  "collection-job-id":
  "<unique_collection_job_name>",
  "collectorType": "SNMP",
  "deviceGroups": [
    {
      "id": "deviceGroup1",
      "devices": [
        {
          "name": "vm1-xrv9000",
          "host": "<device_IP_address>"
        },
        {
          "name": "vm2-xrv9000",
          "host": "<device_IP_address>"
        },
        {
          "name": "vm3-xrv9000",
          "host": "<device_IP_address>"
        }
      ]
    },
    {
      "deviceParams": [
        {
          "key": "snmpVersion",
          "value": "V3"
        },
        {
          "key": "snmpSecurityName",
          "value": "xrvr-usr"
        },
        {
          "key": "snmpAuthProtocol",
          "value": "MD5"
        },
        {
          "key": "snmpAuthPassphrase",
          "value": "<password>"
        },
        {
          "key": "snmpPrivProtocol",
          "value": "AES128"
        },
        {
          "key": "snmpPrivPassphrase",
          "value": "<password>"
        },
        {
          "key": "snmpCommunityString",
          "value": "xrvr-usr"
        }
      ]
    }
  ],
  "sources": [
    {
      "name": "<unique_source_name>",
      "deviceGroupIds": [
        "deviceGroup1"
      ]
    }
  ]
}

```

```

        "sensorConfigs": [
            {
                "sensorConfigId":
"sensor-id-1",
                "path": "1.3.6.1.2.1.1.3.0",
                "operation": "SCALAR",
                "cadence": 60,
                "startTime": 90,
                "collectCount": 10
            },
            {
                "sensorConfigId":
"sensor-id-2",
                "path": "IF-MIB",
                "operation": "MIB_WALK",
                "cadence": 90,
                "startTime": 90,
                "collectCount": 10
            },
            {
                "sensorConfigId":
"sensor-id-3",
                "path": "ifStackTable",
                "operation": "TABLE",
                "cadence": 120,
                "sensorPathParams": [
                    {
                        "key": "MIB",
                        "value": "IF-MIB"
                    }
                ]
            }
        ],
        "sinks": [
            {
                "name": "<unique_sink_name>",
                "parentIds": [
                    "<unique_source_name>"
                ],
                "outputTarget": {
                    "messageBusTarget": {
                        "brokerAddresses": [
                            {
                                "host":
"kafka-operational",
                                "port": 9092
                            }
                        ],
                        "username": "kafka-admin",
                        "password": "<password>",
                        "destinationTopic":
"externalOutputTopic",
                        "certificateKey": "BasicOne"
                    }
                }
            }
        ],
        "secretKey": "skey"
    }

```

SNMP payload example with gRPC as output target

```
"sinks": [
  {
    "name": "<unique_sink_name>",
    "parentIds": [
      "<unique_source_name>"
    ],
    "outputTarget": {
      "grpcTarget": {
        "password": "<password>",
        "username": "admin",
        "certificateKey": "some
certificate",
        "grpcAddress": {
          "port": 60061,
          "host": "<gRPC_server_IP_address>"
        }
      }
    }
  }
],
"secretKey": "skey"
}
```

<p>Kafka Output Target (Custom MIB package)</p> <p>Note The custom MIB package can be imported into the container using Import MIB Pack Tar File option from the Main Menu of Crosswork Data Gateway. See Deploy a Custom MIB, on page 90.</p>	
---	--

```

{
  "collection-job-id":
"<unique_collection_job_name>",
  "collectorType": "SNMP",
  "deviceGroups": [{
    "id": "deviceGroup1",
    "devices": [{
      "name": "vm1-xrv9000",
      "host": "<device_IP_address>"
    },
    {
      "name": "vm2-xrv9000",
      "host": "<device_IP_address>"
    },
    {
      "name": "vm3-xrv9000",
      "host": "<device_IP_address>"
    }
  ]
}],
  "deviceParams": [{
    "key": "snmpVersion",
    "value": "V3"
  },
  {
    "key": "snmpSecurityName",
    "value": "xrvr-usr"
  },
  {
    "key": "snmpAuthProtocol",
    "value": "MD5"
  },
  {
    "key": "snmpAuthPassphrase",
    "value": "<password>"
  },
  {
    "key": "snmpPrivProtocol",
    "value": "AES128"
  },
  {
    "key": "snmpPrivPassphrase",
    "value": "<password>"
  },
  {
    "key": "snmpCommunityString",
    "value": "xrvr-usr"
  }
  ],
  "sources": [{
    "name": "<unique_source_name>",
    "deviceGroupIds": [
      "deviceGroup1"
    ],
    "sensorConfigs": [{
      "sensorConfigId": "sensor-id-1",
      "path": "*",
      "operation": "DEVICE_PACKAGE",
      "cadence": 120,
      "sensorPathParams": [{
        "key": "package",

```

```
        "value": "getIfNumbersnmp"
      },
      {
        "key": "function",
        "value": "SnmpActionIfMib"
      }
    ]
  }
}],
"sinks": [{
  "name": "<unique_sink_name>",
  "parentIds": [
    "<unique_source_name>"
  ],
  "outputTarget": {
    "messageBusTarget": {
      "brokerAddresses": [{
        "host":
" kafka-operational",
        "port": 9092
      }],
      "username": "kafka-admin",
      "password": "<password>",
      "destinationTopic":
" externalOutputTopic",
      "certificateKey": "BasicOne"
    }
  }
}],
"secretKey": "skey"
}
```

gRPC Output Target

(with custom MIB)

```

{
  "collection-job-id":
  "<unique_collection_job_name>",
  "collectorType": "SNMP",
  "deviceGroups": [{
    "id": "deviceGroup1",
    "devices": [{
      "name": "vm1-xrv9000",
      "host": "<device_IP_address>"
    },
    {
      "name": "vm2-xrv9000",
      "host": "<device_IP_address>"
    },
    {
      "name": "vm3-xrv9000",
      "host": "<device_IP_address>"
    }
  ]
}],
  "deviceParams": [{
    "key": "snmpVersion",
    "value": "V3"
  },
  {
    "key": "snmpSecurityName",
    "value": "xrvr-usr"
  },
  {
    "key": "snmpAuthProtocol",
    "value": "MD5"
  },
  {
    "key": "snmpAuthPassphrase",
    "value": "<password>"
  },
  {
    "key": "snmpPrivProtocol",
    "value": "AES128"
  },
  {
    "key": "snmpPrivPassphrase",
    "value": "<password>"
  },
  {
    "key": "snmpCommunityString",
    "value": "xrvr-usr"
  }
  ],
  "sources": [{
    "name": "<unique_source_name>",
    "deviceGroupIds": [
      "deviceGroup1"
    ],
    "sensorConfigs": [{
      "sensorConfigId": "sensor-id-1",
      "path": "*",
      "operation": "DEVICE_PACKAGE",
      "cadence": 120,
      "sensorPathParams": [{
        "key": "package",

```

```

        "value": "getIfNumbersnmp"
    },
    {
        "key": "function",
        "value": "SnmplibActionIfMib"
    }
]
}],
"sinks": [{
    "name": "<unique_sink_name>",
    "parentIds": [
        "<unique_source_name>"
    ],
    "outputTarget": {
        "grpcTarget": {
            "password": "<password>",
            "username": "admin",
            "certificateKey": "some
certificate",
            "grpcAddress": {
                "port": 60061,
                "host": "<gRPC_server_IP_address>"
            }
        }
    }
}],
"secretKey": "skey"
}

```

SNMP Traps Collection Job

SNMP traps are handled in a similar manner. Trap listeners listen on a port and then dispatch data to recipients (based on their topic of interest).



Note

- Device should have been pre-configured by the traps.
- Crosswork Data Gateway listens on UDP port 1062 for Traps.
- Collection jobn gets stuck in Accepted state when wrong authentication credentials are passed for SNMP Traps.

On receiving a trap, Crosswork Data Gateway does the following validations:

1. Check if any collection job is created for the device.
2. Checks the trap version and community string.
3. For SNMP v3, validates for user auth and priv protocol and credentials.

The following table shows different types of SNMP Trap collection jobs:

API URL	{contextPath}/v1/collect-jobs/{jobId}
---------	---------------------------------------

HTTP Method	GET
Response Status Code	200 OK

SNMP TRAP with Kafka output target	
------------------------------------	--

```

{
  "collection-job-id":
  "<unique_collection_job_name>",
  "collectorType": "SNMP",
  "deviceGroups": [
    {
      "id": "deviceGroup1",
      "devices": [
        {
          "name": "vm1-xrv9000",
          "host": "<device_IP_address>"
        },
        {
          "name": "vm2-xrv9000",
          "host": "<device_IP_address>"
        },
        {
          "name": "vm3-xrv9000",
          "host": "<device_IP_address>"
        }
      ]
    }
  ],
  "deviceParams": [
    {
      "key": "snmpVersion",
      "value": "V3"
    },
    {
      "key": "snmpSecurityName",
      "value": "xrvr-usr"
    },
    {
      "key": "snmpAuthProtocol",
      "value": "MD5"
    },
    {
      "key": "snmpAuthPassphrase",
      "value": "<password>"
    },
    {
      "key": "snmpPrivProtocol",
      "value": "AES128"
    },
    {
      "key": "snmpPrivPassphrase",
      "value": "<password>"
    },
    {
      "key": "snmpEngineId",
      "value": "00000009030052540020fd32"
    },
    {
      "key": "snmpCommunityString",
      "value": "xrvr-usr"
    }
  ]
  "sources": [
    {
      "name": "<unique_source_name>",
      "deviceGroupIds": [
        "deviceGroup1"
      ]
    }
  ]
}

```

```
],
"sensorConfigs": [
  {
    "sensorConfigId": "sensor-id-1",
    "operation": "TRAP",
    "path": "*"
  },
  {
    "sensorConfigId": "sensor-id-2",
    "operation": "TRAP",
    "path": "*"
  }
]
},
"sinks": [
  {
    "id": "<unique_sink_name>",
    "name": "sink2",
    "parentIds": [
      "<unique_source_name>"
    ],
    "outputTarget": {
      "messageBusTarget": {
        "brokerAddresses": [
          {
            "host": "kafka-operational",
            "port": 9092
          }
        ],
        "username": "kafka-admin",
        "password": "<password>",
        "destinationTopic":
"externalOutputTopic",
        "certificateKey": "BasicOne"
      }
    }
  }
],
"secretKey": "skey"
}
```

SNMP TRAP with gRPC output target

```

{
  "collection-job-id": "collect-job1",
  "collectorType": "SNMP",
  "deviceGroups": [
    {
      "id": "deviceGroup1",
      "devices": [
        {
          "name": "vm1-xrv9000",
          "host": "<device_IP_address>"
        }
      ],
      "deviceParams": [
        {
          "key": "snmpVersion",
          "value": "V2"
        },
        {
          "key": "snmpCommunityString",
          "value": "xrvr-usr"
        }
      ]
    }
  ],
  "sources": [
    {
      "name": "source1",
      "deviceGroupIds": [
        "deviceGroup1"
      ],
      "sensorConfigs": [
        {
          "path": "*",
          "operation": "TRAP"
        }
      ]
    }
  ],
  "sinks": [
    {
      "name": "sink-grpc",
      "parentIds": [
        "source1"
      ],
      "outputTarget": {
        "grpcTarget": {
          "password": "<password>",
          "username": "admin",
          "certificateKey": "some
certificate",
          "grpcAddress": {
            "port": 60061,
            "host": "<gRPC_server_IP_address>"
          }
        }
      }
    }
  ],
  "secretKey": "skey"
}

```

MDT Collection Job

Crosswork Data Gateway supports data collection from network devices using Model-driven Telemetry (MDT) to consume telemetry streams directly from devices.



Note It is suggested that the Controller application can integrate with a provisioning entity to provision the telemetry subscription request on the device. Crosswork Data Gateway is not responsible to provision the MDT on the device. It is customer's responsibility to perform any device configuration required on the device using some other provisioning tool.

It supports data collection for the following transport modes:

- [MDT gRPC Dial-in Mode, on page 85](#)



Note Dial-in mode is supported only over gRPC.

- [MDT gRPC Dial-out Mode, on page 86](#)
- [MDT TCP Dial-out Mode, on page 87](#)

Source Fields Interpretation for MDT

Field	Description
sensorConfigs->path	Subscription ID for MDT subscriptions on the device.
deviceSet->Name	Field to match with node_id from collected MDT Header data.
deviceSet->transport	MDT collection method: <ul style="list-style-type: none"> • GRPC_DAILIN • GRPC_DAILOUT • TCP_DAILOUT
deviceSet->deviceAddress->host	Host to connect for GRPC_DAILIN
deviceSet->deviceAddress->port	Port to be used for GRPC_DAILIN only
deviceSet->deviceAddress->username deviceSet->deviceAddress->password	Credentials for GRPC_DAILIN only. This can be any user who has access to subscribed configs.



Note Any other fields, if populated, would be ignored by MDT collector.

MDT gRPC Dial-in Mode

```

{
  "collectionJobId": "Mdt_Grpc_DialIn_OneSource_OneSink_OneDevice_OneConfig_Kafka",
  "collectorType": "MDT",
  "deviceGroups": [
    {
      "id": "deviceGroup1",
      "devices": [
        {
          "name": "vm2-xrv9000",
          "host": "<device_IP_address>"
        }
      ],
      "deviceParams": [
        {
          "key": "transport",
          "value": "GRPC_DIALIN"
        },
        {
          "key": "username",
          "value": "admin"
        },
        {
          "key": "password",
          "value": "<password>"
        },
        {
          "key": "port",
          "value": "17777"
        }
      ]
    }
  ],
  "sources": [
    {
      "name": "mdt-grpc-dial-in-source",
      "deviceGroupIds": [
        "deviceGroup1"
      ],
      "sensorConfigs": [
        {
          "sensorConfigId": "5b8d749a-74f2-420a-beea-c0e19a6b7a63",
          "path": "GrpcDialInSub1"
        }
      ]
    }
  ],
  "sinks": [
    {
      "name": "kafka-sink",
      "parentIds": [
        "mdt-grpc-dial-in-source"
      ],
      "outputTarget": {
        "messageBusTarget": {
          "brokerAddresses": [
            {
              "host": "<kafka_server_IP_address>",
              "port": 9092
            }
          ],
          "destinationTopic": "mdt-grpc-dialin-collection-topic"
        }
      }
    }
  ]
}

```

```

    }
  ],
  "secretKey": "skey"
}

```

Sink with gRPC Output Target

```

"sinks": [
  {
    "name": "sink-grpc",
    "parentIds": [
      "source1"
    ],
    "outputTarget": {
      "grpcTarget": {
        "password": "<password>",
        "username": "admin",
        "certificateKey": "some certificate",
        "grpcAddress": {
          "port": 60061,
          "host": "<device_IP_address>"
        }
      }
    }
  }
]

```

MDT gRPC Dial-out Mode

```

{
  "collection-job-id": "Mdt_Grpc_DialOut_OneSource_OneSink_OneDevice_OneConfig",
  "collectorType": "MDT",
  "deviceGroups": [
    {
      "id": "deviceGroup1",
      "devices": [
        {
          "name": "vm2-xrv9000",
          "host": "<device_IP_address>"
        }
      ],
      "deviceParams": [
        {
          "key": "transport",
          "value": "GRPC_DIALOUT"
        }
      ]
    }
  ],
  "sources": [
    {
      "name": "mdt-grpc-dial-out-source",
      "deviceGroupIds": [
        "deviceGroup1"
      ],
      "sensorConfigs": [
        {
          "sensorConfigId": "5b8d749a-74f2-420a-beea-c0e19a6b7a63",
          "path": "Sub1"
        }
      ]
    }
  ]
}

```

```

"sinks": [
  {
    "name": "kafka-sink",
    "parentIds": [
      "mdt-grpc-dial-out-source"
    ],
    "outputTarget": {
      "messageBusTarget": {
        "brokerAddresses": [
          {
            "host": "<kafka_server_IP_address>",
            "port": 9092
          }
        ],
        "destinationTopic": "mdt-collection-topic"
      }
    }
  }
],
"secretKey": "skey"
}

```

Sink with gRPC Output Target

```

"sinks": [
  {
    "name": "sink-grpc",
    "parentIds": [
      "source1"
    ],
    "outputTarget": {
      "grpcTarget": {
        "password": "<password>",
        "username": "admin",
        "certificateKey": "some certificate",
        "grpcAddress": {
          "port": 60061,
          "host": "<gRPC_server_IP_address>"
        }
      }
    }
  }
]

```

MDT TCP Dial-out Mode

```

{
  "collectionJobId": "Mdt_Tcp_DialOut_OneSource_OneSink_OneDevice_OneConfig_kafka",
  "collectorType": "MDT",
  "deviceGroups": [
    {
      "id": "deviceGroup1",
      "devices": [
        {
          "name": "vm2-xrv9000",
          "host": "<device_IP_address>"
        }
      ],
      "deviceParams": [
        {
          "key": "transport",
          "value": "TCP_DIALOUT"
        }
      ]
    }
  ]
}

```

```

    }
  ],
  "sources": [
    {
      "name": "mdt-tcp-dial-out-source",
      "deviceGroupIds": [
        "deviceGroup1"
      ],
      "sensorConfigs": [
        {
          "sensorConfigId": "5b8d749a-74f2-420a-beea-c0e19a6b7a63",
          "path": "TcpSub1"
        }
      ]
    }
  ],
  "sinks": [
    {
      "name": "kafka-sink",
      "parentIds": [
        "mdt-tcp-dial-out-source"
      ],
      "outputTarget": {
        "messageBusTarget": {
          "brokerAddresses": [
            {
              "host": "<gRPC_server_IP_address>",
              "port": 9092
            }
          ],
          "destinationTopic": "mdt-tcp-dialout-collection-topic"
        }
      }
    }
  ],
  "secretKey": "skey"
}

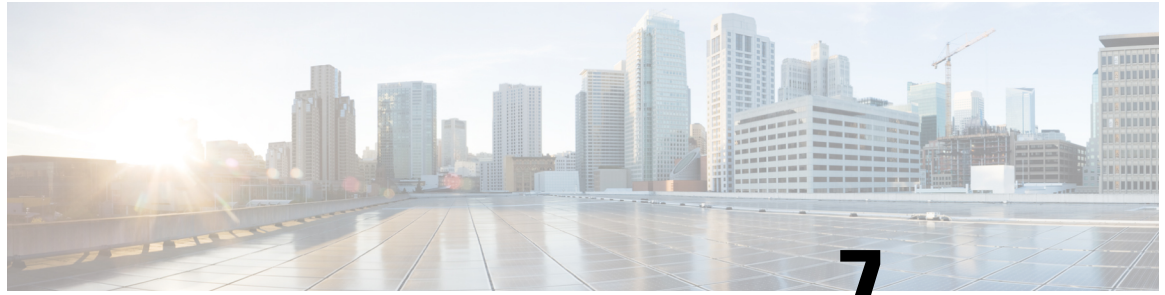
```

Sink with gRPC Output Target

```

"sinks": [
  {
    "name": "sink-grpc",
    "parentIds": [
      "source1"
    ],
    "outputTarget": {
      "grpcTarget": {
        "password": "<password>",
        "username": "admin",
        "certificateKey": "some certificate",
        "grpcAddress": {
          "port": 60061,
          "host": "<device_IP_address>"
        }
      }
    }
  }
]

```



CHAPTER 7

Deploying Custom Packages

To support 3rd party device CLI and SNMP MIBs, Crosswork Data Gateway allows you to import the device packages and MIBs to the collectors. If there is a need to perform any custom transformation, it should be in-built into these packages.

This section contains the following topics:

- [Types of Custom Packages, on page 89](#)
- [Deploy a Custom Device Package, on page 89](#)
- [Deploy a Custom MIB, on page 90](#)

Types of Custom Packages

Crosswork Data Gateway allows you to register and deploy two types of Custom Packages:

1. Custom Device Packages
2. Custom MIBs

Deploy a Custom Device Package

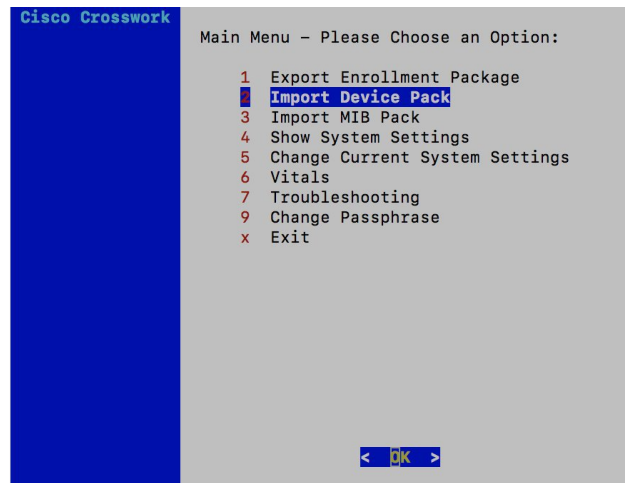
Follow these steps to import a custom device package into Crosswork Data Gateway:



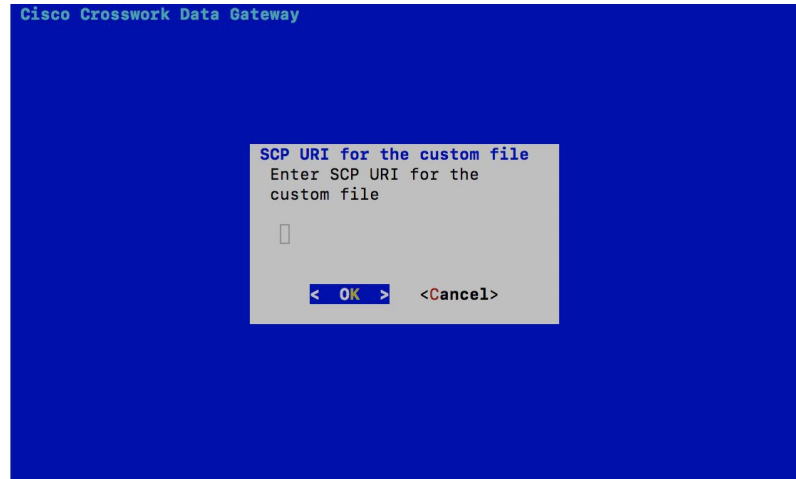
Note Ensure that the custom device package TAR file has just the device package folders and none of the parent folder or hierarchy of folders as part of the TAR file.

If not exported properly, Crosswork Data Gateway throws exceptions when executing the job with custom device package.

Step 1 From the Main Menu, select **2 Import Device Pack Tar File**.



Step 2 Click **OK**. Crosswork Data Gateway prompts you to input SCP URI for the custom device package.



Step 3 Enter the SCP URI and click **OK**.

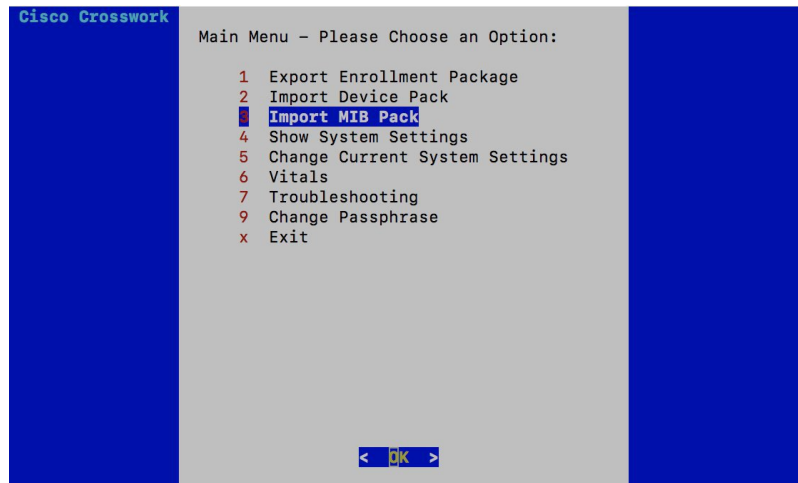
Step 4 Enter the SCP passphrase and click **OK**.

The device package is uploaded to `/opt/dg/custom/device-packages`.

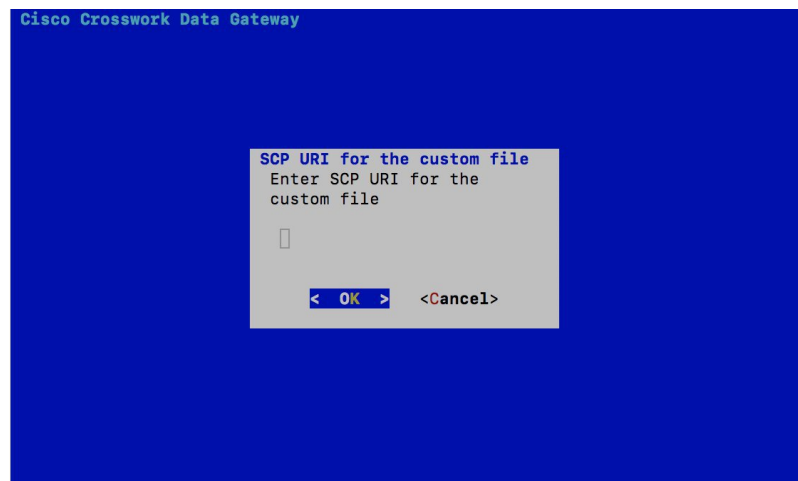
Deploy a Custom MIB

Follow these steps to import a custom MIB into Crosswork Data Gateway:

Step 1 From the Main Menu, select **3 Import MIB Pack Tar File**.



Step 2 Click **OK**. Crosswork Data Gateway prompts you to input SCP URI for the custom MIB package.



Step 3 Enter the SCP URI and click **OK**.

Step 4 Enter the SCP passphrase and click **OK**.

The MIB package is uploaded to `/opt/dg/custom/device-packages`.



CHAPTER 8

Monitor Crosswork Data Gateway Health

This section contains the following topics:

- [Vitals Monitor, on page 93](#)
- [View Crosswork Data Gateway Vitals, on page 93](#)
- [collector-vitals Service, on page 97](#)

Vitals Monitor

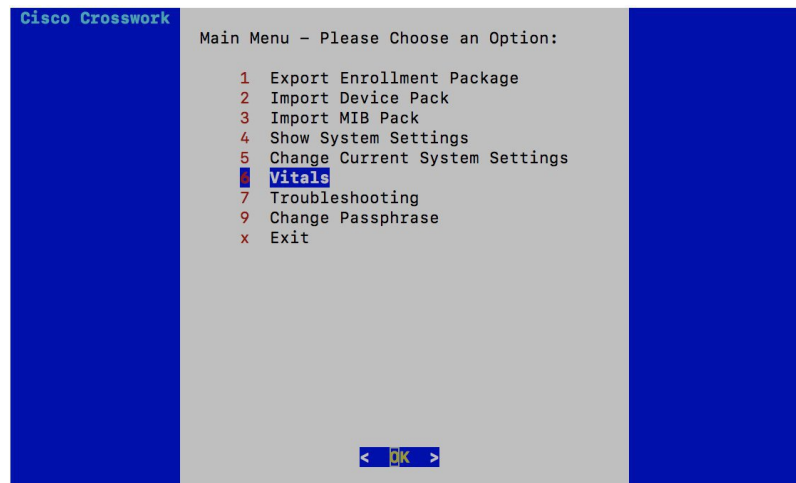
The Vitals Monitor component of Crosswork Data Gateway enables you to view vitals for the following:

1. Docker containers
2. Docker images
3. Controller reachability
4. NTP reachability
5. Route table
6. ARP table
7. Network connections
8. Disk space usage

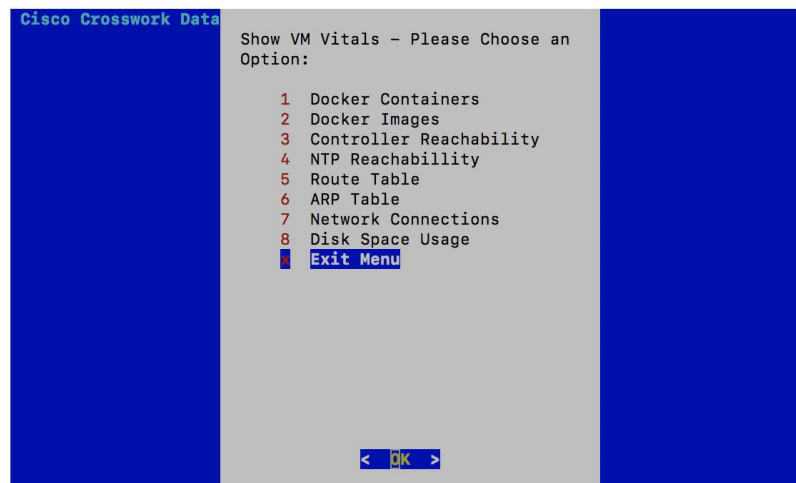
View Crosswork Data Gateway Vitals

Follow these steps to view Crosswork Data Gateway vitals:

Step 1 From the Main Menu, select **6 Vitals**, as shown in the following figure.



Step 2 Click **OK**. The **Show VM Vitals** menu opens.



Step 3 Select the vital you want to view.

Step 4 Click **OK**. Crosswork Data Gateway displays the vitals for the selected item.

After you are done viewing the vitals, press any key to return to the **ShowVM Vitals** menu.

To return to the Main Menu, select **x Exit Menu** and click **OK**.

View Docker Containers Vitals

Follow this procedure to view the following vitals of the Docker containers:

- Container ID
- Image
- Name

- Command
- Created Time
- Status
- Port

Step 1 From the **Show VM Vitals** Menu, select **1 Docker Containers**.

Step 2 Click **OK**. **Docker Containers** vitals are displayed.

View Docker Images Vitals

Follow this procedure to view the following vitals of the Docker images:

- Repository
- Image ID
- Created Time
- Size
- Tag

Step 1 From the **Show VM Vitals** Menu, select **2 Docker Images**.

Step 2 Click **OK**. **Docker Images** vitals are displayed.

View Controller Reachability Vitals

Follow this procedure to view the following Controller reachability vitals:

- Default gateway
- Default gateway status
- Reachability test details (number of packets transmitted and received, packet loss percentage, and time)
- DNS server
- DNS server status
- Reachability test details (number of packets transmitted and received, packet loss percentage, and time)
- Controller session status

Step 1 From the **Show VM Vitals** Menu, select **3 Controller Reachability**.

Step 2 Click **OK**. Crosswork Data Gateway tests the reachability to the Controller and DNS server and displays the vitals.

View NTP Reachability Vitals

Follow this procedure to view the following NTP reachability vitals:

- NTP server
 - Resolved IP Address
 - Status
 - Reachability test details (number of packets transmitted and received, packet loss percentage, and time)
 - Chrony status
 - Reference ID
 - System time
-

Step 1 From the **Show VM Vitals** Menu, select **4 NTP Reachability**.

Step 2 Click **OK**. Crosswork Data Gateway tests the reachability to the NTP server and displays the vitals.

View Route Table

Follow this procedure to view IPv4 and IPv6 route tables.

Step 1 From the **Show VM Vitals** Menu, select **5 Route Table**.

Step 2 Click **OK**. Crosswork Data Gateway displays the route tables.

View ARP Table

Follow this procedure to view ARP tables.

Step 1 From the **Show VM Vitals** Menu, select **6 ARP Table**.

Step 2 Click **OK**. Crosswork Data Gateway displays the ARP tables.

View Network Connections

Follow this procedure to view the following network connections vitals:

- Netid

- State
 - Recv-Q
 - Send-Q
 - Local Address and Port
 - Peer Address and Port
-

- Step 1** From the **Show VM Vitals** Menu, select **7 Network Connections**.
- Step 2** Click **OK**. Crosswork Data Gateway displays the network connections vitals.
-

View Disk Space Usage

Follow this procedure to view disk space usage:

- Filesystem
 - Size
 - Used space
 - Available space
 - Use percentage
 - Mounted on volume
-

- Step 1** From the **Show VM Vitals** Menu, select **8 Disk Space Usage**.
- Step 2** Click **OK**. Crosswork Data Gateway displays the disk space usage.
-

collector-vitals Service

Crosswork Data Gateway comprises of various containerized services running on an Ubuntu VM. Its overall health depends on health of each containerized service.

As part of collector vitals, Crosswork Data Gateway collects host and container metrics and writes them to a container mounted path in **vitals.json** file and sends it to the Controller.

It collects the following metrics:

Crosswork Data Gateway Host VM/Component	Metric	Description
Crosswork Data Gateway Host VM	CPU	CPU consumption - amount of actively used CPU.
	Memory	Disk Usage (in GB) - used memory indicates used+buff/cache values.
	Load	The average system load calculated over a given period of time of 1, 5 and 15 minutes.
	Disk I/O	Disk I/O includes read/write or input/output operations (defined in MB/s) involving a disk.
	Network I/O	The amount of data sent/received in MB for NIC interfaces - eth0, eth1, and eth2.
Crosswork Data Gateway Components	Status	Running (Liveness - criteria), Stopped, or Not Deployed
	CPU %	CPU consumption - the percentage of the host's CPU the component container is using.
	Memory Usage/Limit	Total memory the container is using and the total amount of memory it is allowed to use.
	Memory %	Percentage of the host's memory the container is using.
	Block I/O	Includes read/write or input/output operations (defined in MB/s) involving a disk.
	Network I/O	The amount of data the component has sent and received over its network interface (in MegaBytes).

**Note**

-
- When the Docker engine or serviced components listed below are not responsive, Crosswork Data Gateway vitals are not updated:

- Vitals Monitor
- Controller Gateway

The "Collector Vitals" and "Controller Gateway" dockers must be up and running for alerts/vitals to get updated.

- When Vitals Monitor stops, no alerts are added to cdg-alerts.log. This is because the monitor service runs as a part of Vitals Monitors and it doesn't trigger any alerts when Vitals Monitor itself is down.
 - Also, the alerts are not added to cdg-alerts.log when Vitals Monitor is running and Controller Gateway is down.
-



CHAPTER 9

Troubleshooting

This section contains the following topics:

- [Troubleshoot Crosswork Data Gateway, on page 101](#)
- [Start/Stop Docker Containers, on page 108](#)

Troubleshoot Crosswork Data Gateway

You can troubleshoot a Crosswork Data Gateway instance either directly from the VM or remotely. Crosswork Data Gateway provides logs of errors, requests to the server, and changes made to the VM and reports any process failures/outages.

To access **Troubleshooting** menu, select **7 Troubleshooting** from the Main Menu and click **OK**, as shown in the following figure:

```
Main Menu - Please Choose an Option:

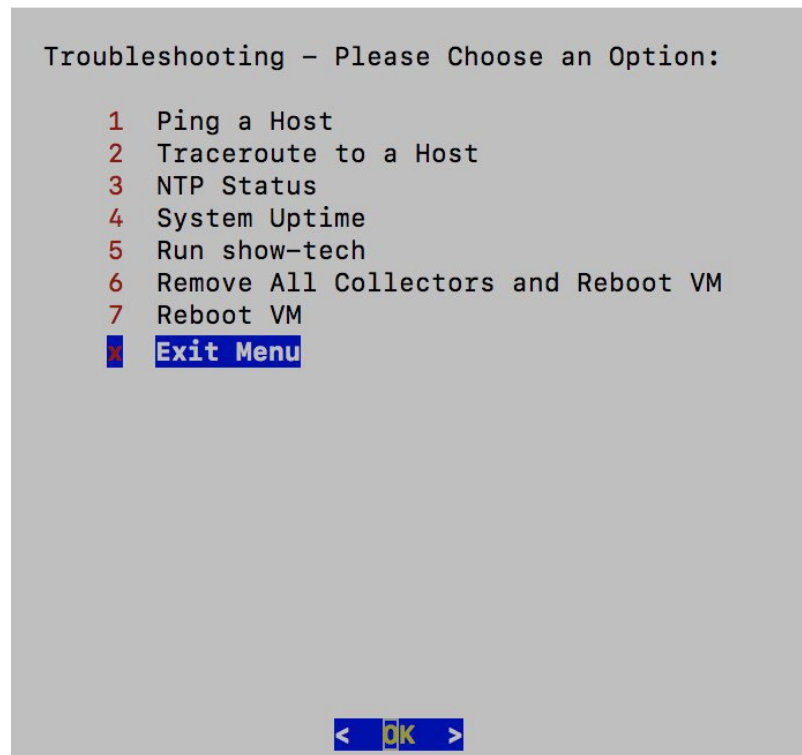
 1 Export Enrollment Package
 2 Import Device Pack Tar File
 3 Import MIB Pack Tar File
 4 Show System Settings
 5 Change Current System Settings
 6 Vitals
 7 Troubleshooting
 p Change Passphrase
 l Logout

< OK >
```

Crosswork Data Gateway opens the **Troubleshooting** Menu that provides you the following options to troubleshoot your Crosswork Data Gateway instance:



Note The following figure shows the Troubleshooting Menu corresponding to **dg-admin** user. Few of these options are not available to **dg-oper** user. See Table [Table 1: Permissions Per Role, on page 29](#).

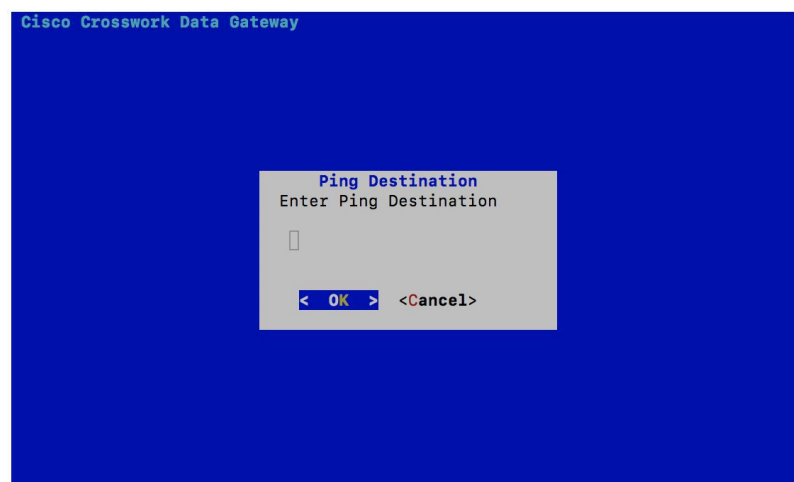


- [Ping a Host, on page 103](#)
- [Traceroute to a Host, on page 104](#)
- [Check NTP Status, on page 105](#)
- [Check System Uptime, on page 106](#)
- [Run show-tech, on page 106](#)
- [Reboot Crosswork Data Gateway VM, on page 107](#)

Ping a Host

To aid troubleshooting, Crosswork Data Gateway provides you Ping utility that can be executed to any IP address to check its reachability.

-
- Step 1** From **Troubleshooting** menu, select **1 Ping a Host**.
- Step 2** Click **OK**. The Crosswork Data Gateway prompts you to enter the ping destination.



Step 3 Enter the ping destination and click **OK**.

Crosswork Data Gateway displays the result of the ping operation.

```

PING 172.23.92.143 (172.23.92.143) 56(84) bytes of data.
64 bytes from 172.23.92.143: icmp_seq=1 ttl=64 time=0.428 ms
64 bytes from 172.23.92.143: icmp_seq=2 ttl=64 time=0.368 ms
64 bytes from 172.23.92.143: icmp_seq=3 ttl=64 time=0.270 ms

64 bytes from 172.23.92.143: icmp_seq=4 ttl=64 time=0.574 ms

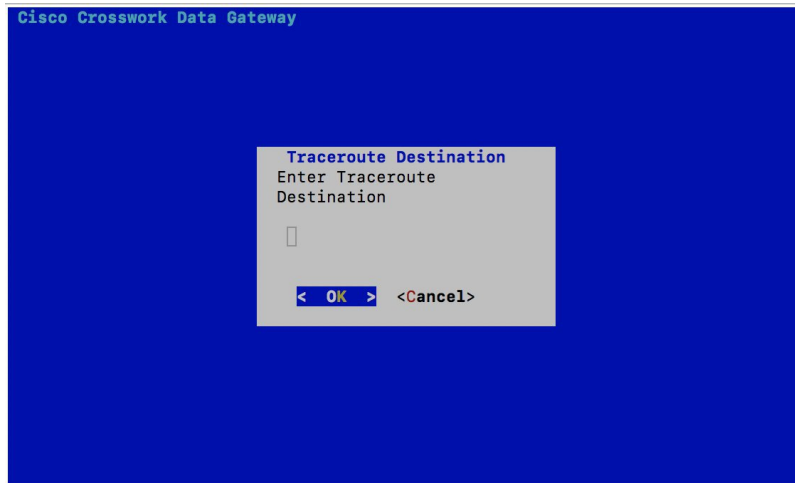
64 bytes from 172.23.92.143: icmp_seq=5 ttl=64 time=0.433 ms
64 bytes from 172.23.92.143: icmp_seq=6 ttl=64 time=0.487 ms
^C
--- 172.23.92.143 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5107ms
rtt min/avg/max/mdev = 0.270/0.426/0.574/0.097 ms
Press any key to continue
  
```

Traceroute to a Host

Crosswork Data Gateway provides **Traceroute to a Host** option to help troubleshoot latency issues. Using this option provides you a rough time estimate for the Crosswork Data Gateway to reach the Controller.

Step 1 From **Troubleshooting** menu, select **2 Traceroute to a Host**. Click **OK**.

Step 2 Enter the traceroute destination and click **OK**.



Step 3 Enter your SCP passphrase and click **OK**.



Check NTP Status

Use this option to check the status of the NTP server.

- Step 1** From **Troubleshooting** menu, select **3 NTP Status**.
- Step 2** Click **OK**. The Crosswork Data Gateway displays the NTP server status.

```

Reference ID   : AB442641 (mtv5-ai27-dcm10n-ntp1.cisco.com)
Stratum       : 2
Ref time (UTC) : Fri Jun 21 04:53:44 2019
System time   : 0.00044881 seconds fast of NTP time
Last offset   : +0.00057586 seconds
RMS offset    : 0.00080841 seconds
Frequency     : 21.559 ppm slow
Residual freq : +0.009 ppm
Skew          : 0.144 ppm
Root delay    : 0.002095408 seconds
Root dispersion : 0.001190380 seconds
Update interval : 2062.6 seconds
Leap status   : Normal
Press any key to continue

```

Check System Uptime

Use this option to check system uptime.

Step 1 From **Troubleshooting** menu, select **4 System Uptime**.

Step 2 Click **OK**. The Crosswork Data Gateway displays the system uptime.

```

05:11:55 up 3 days, 1:49, 1 user, load average: 0.18, 0.12, 0.10
Press any key to continue

```

Run show-tech

Crosswork Data Gateway provides the option **show_tech** to gather its state data.

The collected data includes the following:

- Logs of all the CDG components running on docker containers
- VM Vitals

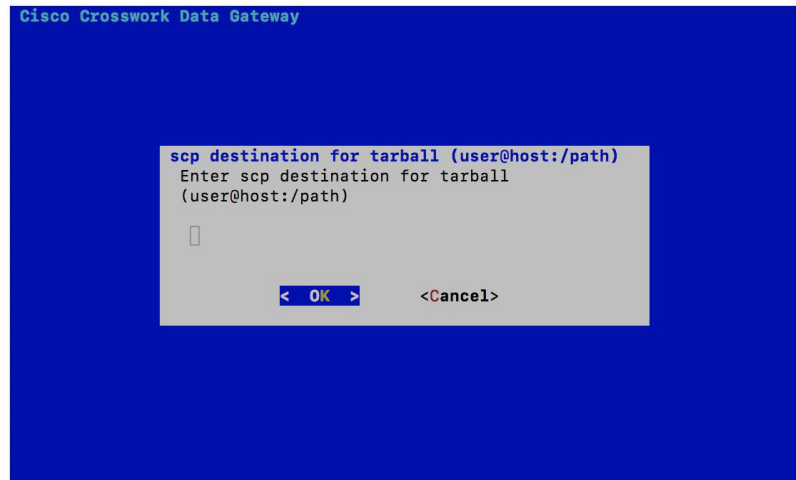
It creates a tarball in the directory where it is executed. The output is a tarball named

```
CDG-<CDG-version>-year-month-day--hour-minute-second-*.tar.bz2
```

The execution of this command may take several minutes depending on the state of Crosswork Data Gateway.

Step 1 From **Troubleshooting** menu, select **5 Show-tech**.

Step 2 Click **OK**. The Crosswork Data Gateway prompts you to enter the destination for the tarball.



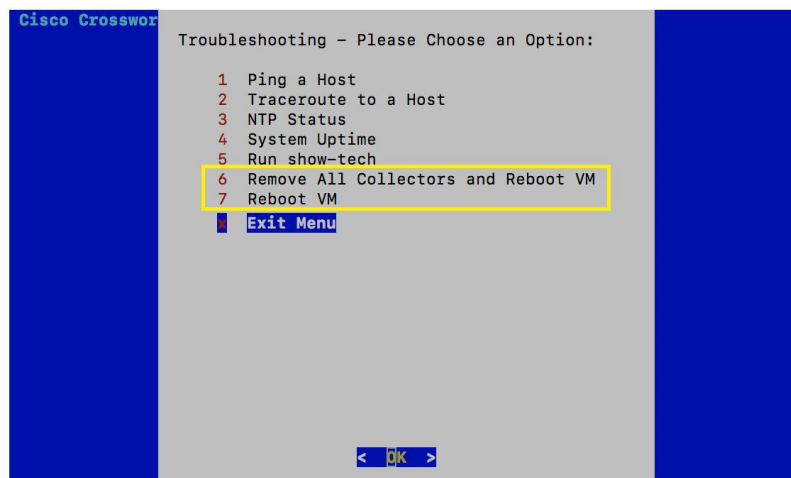
Step 3 Enter the destination for the tarball and click **OK**.
Crosswork Data Gateway displays its state data.

Reboot Crosswork Data Gateway VM



Note This task can only be performed by **dg-admin** user.

Crosswork Data Gateway gives you two options to reboot the VM:



- **Remove All Collectors and Reboot VM:** Select this option from the **Troubleshooting** menu if you want to remove all the collectors (functional images) and reboot VM.
- **Reboot VM:** Select this option from the **Troubleshooting** menu for a normal reboot.



Note During the Crosswork Data Gateway VM reboot, all system containers are in the process of being initialized, hence no CDG_CONTAINER_STATE_CHANGE alert regarding startup of the containers is sent to Controller. The CDG_CONTAINER_STATE_CHANGE alert is sent only when all containers are up and running and a container stops or starts corresponding.

Below is an example of CDG_CONTAINER_STATE_CHANGE alert sent to Controller on stop of a container:

```
Time: 2019-07-13T11:28:37.68316, Data: {"code":"CDG_CONTAINER_STATE_CHANGE",
"time":"1563017317677","level":"LOG_CRIT","containerVitalEvent":{"containerId":
"26408f9c59a5","containerName":"image-manager","state":"STOPPED","message":
"image-manager has exited"}}
```

Below is an example of CDG_CONTAINER_STATE_CHANGE alert sent to Controller on start of a container:

```
Time: 2019-07-13T21:11:38.599944, Data: {"code":"CDG_CONTAINER_STATE_CHANGE",
"time":"1562620298329","level":"LOG_CRIT","containerVitalEvent":{"containerId":
"2d5e845d656a","containerName":"snmp-collector","state":"STARTED","message":
"snmp-collector has started"}}
```

The above two alerts are not sent to Controller during reboot.

Start/Stop Docker Containers

Crosswork Data Gateway does not support manual start/stop of the docker containers.

Run the following command to start the containers:

```
service dg-docker start
```

Run the following command to stop the containers:

```
service dg-docker stop
```