Cisco Spaces: Connector3 Command Reference Guide

First Published: 2022-09-12

Last Modified: 2024-02-14

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

© 2024 Cisco Systems, Inc. All rights reserved.

CONTENTS

Preface

Audience

This document is meant for Cisco Spaces network and IT administrators who deploy Cisco Spaces to monitor, manage, and optimize usage of assets in an organization.

Conventions

This document uses the following conventions.

Table 1: Conventions

Convention	Indication
bold font	Commands and keywords and user-entered text appear in **bold** font.
italic font	Document titles, new or emphasized terms, and arguments for which you supply values are in *italic* font.
[ ]	Elements in square brackets are optional.
{x | y | z }	Required alternative keywords are grouped in braces and separated by vertical bars.
[x | y | z]	Optional alternative keywords are grouped in brackets and separated by vertical bars.
string	A nonquoted set of characters. Do not use quotation marks around the string. Otherwise, the string will include the quotation marks.
`courier` font	Terminal sessions and information the system displays appear in `courier` font.
<>	Nonprinting characters such as passwords are in angle brackets.
[]	Default responses to system prompts are in square brackets.

Convention	Indication
!, #	An exclamation point (!) or a pound sign (#) at the beginning of a line of code indicates a comment line.

Note Means reader take note. Notes contain helpful suggestions or references to material not covered in the manual.

Tip Means the following information will help you solve a problem.

Caution Means reader be careful. In this situation, you might perform an action that could result in equipment damage or loss of data.

Related Documentation

Cisco Spaces: Connector3 Configuration Guide

Cisco Spaces: Connector3 Command Reference Guide

Release Notes for Cisco Spaces: Connector

Cisco Spaces: IoT Service Configuration Guide (Wireless)

Cisco Spaces: IoT Service Configuration Guide (Wired)

Communications, Services, and Additional Information

- To receive timely, relevant information from Cisco, sign up at Cisco Profile Manager.
- To get the business impact you're looking for with the technologies that matter, visit Cisco Services.
- To submit a service request, visit Cisco Support.
- To discover and browse secure, validated enterprise-class apps, products, solutions, and services, visit Cisco DevNet.
- To obtain general networking, training, and certification titles, visit Cisco Press.
- To find warranty information for a specific product or product family, access Cisco Warranty Finder.

Cisco Bug Search Tool

Cisco Bug Search Tool (BST) is a gateway to the Cisco bug-tracking system, which maintains a comprehensive list of defects and vulnerabilities in Cisco products and software. The BST provides you with detailed defect information about your products and software.

Documentation Feedback

To provide feedback about Cisco technical documentation, use the feedback form available in the right pane of every online document.

Restricted Command-Line Interface

Restricted CLI

In Cisco Spaces: Connector, Linux commands are restricted to prevent unauthorized users from inadvertently modifying the system configurations. This restricted access prevents users from modifying system configuration that are likely to cause issues.

The following commands are allowed on the restricted command line.

Table 2: List of Restricted Commands

Command	Description
cat	Prints file contents.
cp	Copies file.
df	Prints file system disk space usage.
du	Prints file space usage.
grep	Prints lines matching a pattern.
ip	Displays network interface configurations.
ls	Lists directory contents.
nslookup	Queries Internet-name servers.
passwd	Changes the **spacesadmin** password.
ping	Sends Internet Control Message Protocol (ICMP) echo requests to network devices.
pwd	Prints the current or working directory.
rm	Removes files.
scp	Secures remote copy files.
sftp	Secures file transfer.
ssh	Connects to cliens with SSH.
tail	Outputs the last part of a file.
top	Displays Linux processes.
route	Configures IP routing table rules.
clear	Clears the screen.
wget	Downloads files from the internet.
who	Displays the user.

Upgrade Commands

connectoros upgrade

To upgrade the installed connector, use the **connectorosupgrade** command.

connectoros upgrade .*connector-image*

Syntax Description

Keywords and Variables	Description
.*connector-image*	.*connector* image downloaded from Cisco.com.

Command History

Release 3	This command is introduced.

Examples

The following is a sample output of the command:

```
[spacesadmin@connector ~]$ connectoros upgrade cisco-dna-spaces-connector-v3.0-.connector
```

Timezone Commands

connectorctl timezone

To manage the system time zone, use the **connectorctl timezone** command.

connectorctl timezone { **-l** | **-s** | **-t** *timezone* }

Syntax Description

Keywords and Variable	Descriptions
-l	Lists all the time zones.
-t *timezone*	Configures one of the timezones. You can list all timezones using the **-l** keyword.
-s	Shows the system time zone.

Command History

Release 3	This command is introduced.

NTP Commands

The NTP commands are not supported on connector AMI.

connectorctl ntp config

To configure the Network Time Protocol (NTP) server, use the **connectorctl ntp config** command.

connectorctl ntp config { **-n** *comma-separated-list-of-servers* | **-d** }

Syntax Description

Keywords and Variables	Description
-n *comma-separated-list-of-servers*	List of NTP servers. Ensure that you separate each server name with a comma.
-d	Deletes the current NTP configurations.

Command History

Release 3	This command is introduced.

Examples

The following example shows how to configure the NTP server:

```
[spacesadmin@connector ~]$ connectorctl ntp config -n ntp.esl.cisco.com
Executing command:ntp
Command execution status:Success
-----------------------
Doing NTP configuration
Checking status for server: ntp.esl.cisco.com
Status check successful for server: ntp.esl.cisco.com
NTP configuration: success
```

connectorctl ntp show

This command shows the Network Time Protocol (NTP) server.

connectorctl ntp show

Syntax Description

This command has no keywords or arguments.

Command History

Release 3	This command is introduced.

Examples

The following is a sample output of the command:

```
[spacesadmin@connector ~]$ connectorctl ntp show
Executing command:ntp
Command execution status:Success
----------------------
==================chrony conf output=================
server ntp.esl.cisco.com
========================end=========================
```

connectorctl ntp status

To observe the status of chrony or Network Time Protocol (NTP) service, sources details, and NTP data details, use the **connectorctl ntp restart** command.

connectorctl ntp status

Syntax Description

This command has no keywords or arguments.

Command History

Release 3	This command is introduced.

Examples

The following is a sample output of the command:

```
[spacesadmin@connector ~]$ connectorctl network status
Executing command:ntp
Command execution status:Success
-----------------------
==================chrony service status=================
chronyd.service - NTP client/server
   Loaded: loaded (/usr/lib/systemd/system/chronyd.service; enabled; vendor preset: enabled)

   Active: active (running) since Thu 2022-07-28 12:20:58 PDT; 5 days ago
     Docs: man:chronyd(8)
           man:chrony.conf(5)
  Process: 895 ExecStartPost=/usr/libexec/chrony-helper update-daemon (code=exited,
status=0/SUCCESS)
  Process: 871 ExecStart=/usr/sbin/chronyd $OPTIONS (code=exited, status=0/SUCCESS)
 Main PID: 877 (chronyd)
    Tasks: 1 (limit: 24285)
   Memory: 2.3M
   CGroup: /system.slice/chronyd.service
           └─877 /usr/sbin/chronyd

Jul 28 12:20:55 conn3-la61-212-23 systemd[1]: Starting NTP client/server...
Jul 28 12:20:56 conn3-la61-212-23 chronyd[877]: chronyd version 4.1 starting (+CMDMON +NTP
 +REFCLOCK +RTC +PRIVDROP +SCFILTER +SIGND +ASYNCDNS +NTS +SECHASH +IPV6 +DEBUG)
Jul 28 12:20:56 conn3-la61-212-23 chronyd[877]: Frequency 0.000 +/- 1000000.000 ppm read
from /var/lib/chrony/drift
Jul 28 12:20:56 conn3-la61-212-23 chronyd[877]: Using right/UTC timezone to obtain leap
second data
Jul 28 12:20:58 conn3-la61-212-23 systemd[1]: Started NTP client/server.
Jul 28 12:23:21 conn3-la61-212-23 chronyd[877]: Selected source 10.68.38.66
(ntp.esl.cisco.com)
Jul 28 12:23:21 conn3-la61-212-23 chronyd[877]: System clock wrong by 1611.296985 seconds
Jul 28 12:50:12 conn3-la61-212-23 chronyd[877]: System clock was stepped by 1611.296985
seconds
Jul 28 12:50:12 conn3-la61-212-23 chronyd[877]: System clock TAI offset set to 37 seconds
==========================end==========================
==================chrony sources==================
MS Name/IP address         Stratum Poll Reach LastRx Last sample
===============================================================================
^* sjc05-73a-dci06n-ntp2.ci>     1  10   377   501    +26us[  +24us] +/-  519us
======================end======================
==================chrony ntpdata==================

Remote address  : 10.68.38.66 (AB442642)
Remote port     : 123
Local address   : 10.22.212.23 (0A16D417)
```

```
Leap status     : Normal
Version         : 4
Mode            : Server
Stratum         : 1
Poll interval   : 10 (1024 seconds)
Precision       : -19 (0.000001907 seconds)
Root delay      : 0.000000 seconds
Root dispersion : 0.000000 seconds
Reference ID    : 474E5353 (GNSS)
Reference time  : Wed Aug 03 04:54:44 2022
Offset          : -0.000023560 seconds
Peer delay      : 0.001034838 seconds
Peer dispersion : 0.000001973 seconds
Response time   : 0.001528190 seconds
Jitter asymmetry: +0.00
NTP tests       : 111 111 1111
Interleaved     : No
Authenticated   : No
TX timestamping : Kernel
RX timestamping : Kernel
Total TX        : 505
Total RX        : 505
Total valid RX  : 505
======================end=====================
```

connectorctl ntp restart

To restart the chrony or the Network Time Protocol (NTP) server, use the **connectorctl ntp restart** command.

connectorctl ntp restart

Syntax Description

This command has no keywords or arguments.

Command History

Release 3	This command is introduced.

Examples

The following is a sample output of the command:

```
[spacesadmin@connector ~]$ connectorctl ntp restart
Executing command:ntp
Command execution status:Success
-----------------------
Restarted ntp/chronyd service
```

Network

The Network commands are not supported on connector AMI.

connectorctl network config

To configure the network, use the **connectorctl network config** command.

connectorctl network config { **-n** *hostname* | **-i** *ip-address* | **-m** *netmask* | **-g** *gateway* | **-o** *domain* | **-n** *interface-name* | **-d** *dns-servers* }

Syntax Description

Keywords and Variables	Description
-n *hostname*	Host name for machine
-i *ip-address*	IPv4 or IPv6 address
-m *netmask*	Netmask for IPv4 or prefix length IPv6
-g *gateway*	Gateway address (IPv4 or IPv6)
-o *domain*	Search domain name
-d *dns-servers*	Comma-separated IP (IPv4 or IPv6) address list for multiple servers
-n *interface-name*	Interface name. Use one of the following: • PRIMARY • SECONDARY

Command History

Release 3	This command is introduced.

Usage Guidelines

You can configure various settings for the network by specifying the right parameters. You can use this command to reconfigure the network entities in case of network change or network disruption.

Caution After you complete the network configuration, the system is automatically rebooted. After this, you lose connectivity and are logged out of the connector GUI.

Examples

The following example shows how to configure the SECONDARY interface on an IPv4 stack, with details such as IP address, stack, search domain name, and DNS:

```
[spacesadmin@connector ~]$ connectorctl network config  -p ipv4 -i 10.7.0.11/24 -g 10.7.0.1
 -o test.com -d 192.168.168.183 -n SECONDARY
Executing command:network
Command execution status:Success
-----------------------
Connection SECONDARY (5e970417-13b4-4ad8-af12-d125ce407c49) successfully added.
Network setup completed with given configuration.
Secondary interface - Added routes.
Secondary interface - Configured firewall zone.
System reboot will happen in 10 seconds. Do not execute any other command...
```

Examples

The following example shows how to configure the SECONDARY interface on an IPv6 stack, with details such as IP address, stack, search domain name, and DNS:

```
[spacesadmin@connector ~]$ connectorctl network config -p ipv6 -i 2001:DB8:303:2021::210/64
 -g 2001:DB8:303:2021::1 -o test.com -d 2001:DB8:68d:4001::a -n SECONDARY

Executing command:network
Command execution status:Success
-----------------------
Connection 'SECONDARY' (5e970417-13b4-4ad8-af12-d125ce407c49) successfully added.
Network setup completed with given configuration.
Secondary interface - Added routes.
Secondary interface - Configured firewall zone.
System reboot will happen in 10 seconds. Do not execute any other command...
```

connectorctl network show

To view the current network configuration and information about primary and secondary interfaces, use the **connectorctl network show** command. To view details of individual interface network, use the **-n** keyword.

connectorctl network show -n *interface-name*

Command History

Release 3	This command is introduced.

Examples

The following example shows how to display network configurations on an IPv4 stack.

```
[spacesadmin@connector ~]$ connectorctl network show
Executing command:network
Command execution status:Success
-----------------------
=================Network Config=================
BOOTPROTO=none
IPADDR=10.22.212.23
NETMASK=255.255.255.0
GATEWAY=10.22.212.1
DNS1=192.70.168.183
DOMAIN=test.com
HWADDR=00:50:56:90:7a:ff
TYPE=Ethernet
PROXY_METHOD=none
BROWSER_ONLY=no
==================end================

[spacesadmin@connector ~]$ connectorctl network show
 Executing command:network
Command execution status:Success
-----------------------
=================Network Config=================
Hostname    - connector-p84-april1


Interface   - PRIMARY
---------------------------------

Network configuration for stack:ipv4
Ip Address  - 10.22.244.180/24
Mac Address - 00:0C:29:EE:24:8A
Gateway     - 10.22.244.1
Dns         - 192.168.168.183
Domain      - test.com


Interface   - SECONDARY
---------------------------------

Network configuration for stack:ipv4
Ip Address  - 7.7.0.11/24
Mac Address - 00:0C:29:EE:24:94
Gateway     - 7.7.0.1
Dns         - 192.168.168.183
Domain      - test.com

==================end================
```

The following example shows how to display only the PRIMARY interface on an IPv4 stack.

```
[spacesadmin@connector ~]$ connectorctl network show -n PRIMARY
Executing command:network
Command execution status:Success
----------------------
=================Network Config=================
Hostname    - connector-p84-april1


Interface   - PRIMARY
--------------------------------

Network configuration for stack:ipv4
Ip Address  - 10.22.244.180/24
Mac Address - 00:0C:29:EE:24:8A
Gateway     - 10.22.244.1
Dns         - 192.168.168.183
Domain      - test.com

=================end================
```

The following example shows how to display only the SECONDARY interface on an IPv4 stack.

```
[spacesadmin@connector ~]$ connectorctl network show -n SECONDARY
Executing command:network
Command execution status:Success
----------------------
=================Network Config=================
Hostname    - dualInt-HA-sec


Interface   - SECONDARY
--------------------------------

Network configuration for stack:ipv4
Ip Address  - 7.7.0.21/24
Mac Address - 00:0C:29:D6:E4:D7
Gateway     - 7.7.0.1
Dns         - 192.168.168.183
Domain      - test.com

=================end================
```

Note The above example assumes the following:

- The PRIMARY interface of the connector is on the 10.22.x.x subnet, and is used to communicate with Cisco Spaces

  .
- The SECONDARY interface of the connector is on the 7.7.x.x subnet, and is used to communicate with all the devices, such as wireless controllers, switches, and APs.

Examples

The following example shows how to display network configurations on an IPv6 stack.

```
[spacesadmin@connector ~]$ connectorctl network show
Executing command:network
Command execution status:Success
----------------------
```

```
=================Network Config=================
Hostname    - conn3-dual-ipv6-p84


Interface   - PRIMARY
---------------------------------

Network configuration for stack:ipv6
Ip Address  - 2001:DB8:303:2022::60/64
Mac Address - 00:0C:29:70:9C:05
Gateway     - 2001:DB8:303:2022::1
Dns         - 2001:DB8:68d:4001::a
Domain      - test.com


Interface   - SECONDARY
---------------------------------

Network configuration for stack:ipv6
Ip Address  - 2001:DB8:303:2021::210/64
Mac Address - 00:0C:29:70:9C:OF
Gateway     - 2001:DB8:303:2021::1
Dns         - 2001:DB8:68d:4001::a
Domain      - test.com

=================end=================
```

The following example shows how to display the PRIMARY interface on an IPv6 stack.

```
[spacesadmin@connector ~]$ connectorctl network show -n PRIMARY
Executing command:network
Command execution status:Success
-----------------------
=================Network Config=================
Hostname    - conn3-dual-ipv6-p84


Interface   - PRIMARY
---------------------------------

Network configuration for stack:ipv6
Ip Address  - 2001:DB8:303:2022::60/64
Mac Address - 00:0C:29:70:9C:05
Gateway     - 2001:DB8:303:2022::1
Dns         - 2001:DB8:68d:4001::a
Domain      - test.com
```

Note The above example assumes the following:

- The PRIMARY interface of the connector is on the 2001:DB8:303:2022::0/64 subnet, and is used to communicate with Cisco Spaces.
- The SECONDARY interface of the connector is on the 2001:DB8:303:2021::0/64 subnet, and is used to communicate with all the devices, such as wireless controllers, switches, and APs.

connectorctl network status

To view the detailed status of the network connectivity of the local machine to the gateway and DNS servers, use the **connectorctl network status** command. This status includes information about both the interfaces. To view the status of individual interface network, using the **--n** keyword.

connectorctl network status -n *interface-name*

Command History

Release 3	This command is introduced.

Examples

The following example shows how to display network connectivity of the local machine to the gateway and DNS servers on an IPv4 stack.

```
[spacesadmin@connector ~]$ connectorctl network status
Executing command:network
Command execution status:Success
-----------------------
Checking connection to 127.0.0.1
Connection check to 127.0.0.1: Success

Checking connection to ip address: 10.22.212.23
Connection check to ip address: Success

Checking connection to gateway: 10.22.212.1
Connection check to gateway: Success

Checking dns connection
Checking dns server resolution: 192.168.168.183
    Status check to dns server 192.168.168.183:
Success

Network status check successful.
```

The following example shows how to display network connectivity on an IPv4 stack configured with dual interface.

```
[spacesadmin@connector ~]$ connectorctl network status
Executing command:network
Command execution status:Success
-----------------------
=================Network Status=================

Network interface - PRIMARY
---------------------------------------

Checking connection status for network stack:ipv4
Checking connection to 127.0.0.1
Connection check to local:127.0.0.1 - Success
Checking connection to 10.22.244.180
Connection check to ip address:10.22.244.180 - Success
Checking connection to 10.22.244.1
Connection check to gateway:10.22.244.1 - Success
Checking dns server resolution: 192.168.168.183
    Status check to dns server 192.168.168.183 - Success
Network interface - PRIMARY status check successful.
```

```
Network interface - SECONDARY
--------------------------------------

Checking connection status for network stack:ipv4
Checking connection to 127.0.0.1
Connection check to local:127.0.0.1 - Success
Checking connection to 7.7.0.11
Connection check to ip address:7.7.0.11 - Success
Checking connection to 7.7.0.1
Connection check to gateway:7.7.0.1 - Success
Checking dns server resolution: 192.168.168.183
    Status check to dns server 192.168.168.183 - Success
Network interface - SECONDARY status check successful.
```

The following example shows how to display the network connectivity details of the PRIMARY interface.

```
[spacesadmin@connector ~]$ connectorctl network status -n PRIMARY
Executing command:network
Command execution status:Success
----------------------
=================Network Status=================

Network interface - PRIMARY
--------------------------------------

Checking connection status for network stack:ipv4
Checking connection to 127.0.0.1
Connection check to local:127.0.0.1 - Success
Checking connection to 10.22.244.180
Connection check to ip address:10.22.244.180 - Success
Checking connection to 10.22.244.1
Connection check to gateway:10.22.244.1 - Success
Checking dns server resolution: 192.168.168.183
    Status check to dns server 192.168.168.183 - Success
Network interface - PRIMARY status check successful.
```

The following example shows how to display the network connectivity details of the SECONDARY interface.

```
[spacesadmin@connector ~]$ connectorctl network status -n SECONDARY
Executing command:network
Command execution status:Success
----------------------
=================Network Status=================

Network interface - SECONDARY
--------------------------------------

Checking connection status for network stack:ipv4
Checking connection to 127.0.0.1
Connection check to local:127.0.0.1 - Success
Checking connection to 7.7.0.11
Connection check to ip address:7.7.0.11 - Success
Checking connection to 7.7.0.1
Connection check to gateway:7.7.0.1 - Success
Checking dns server resolution: 192.168.168.183
    Status check to dns server 192.168.168.183 - Success
Network interface - SECONDARY status check successful.
```

Note The above example assumes the following:

- The PRIMARY interface of the connector is on the 10.22.x.x subnet, and is used to communicate with Cisco Spaces

  .
- The SECONDARY interface of the connector is on the 7.7.x.x subnet, and is used to communicate with all the devices, such as wireless controllers, switches, and APs.

Examples

The following example shows how to display status of network connectivity of the local machine to the gateway and DNS servers on an IPv6 stack configured with a dual interface.

```
[spacesadmin@connector ~]$ connectorctl network status
Executing command:network
Command execution status:Success
----------------------
=================Network Status=================

Network interface - PRIMARY
---------------------------------------

Checking connection status for network stack:ipv6
Checking connection to ::1
Connection check to local:::1 - Success
Checking connection to 2001:DB8:303:2022::60
Connection check to ip address:2001:DB8:303:2022::60 - Success
Checking connection to 2001:DB8:303:2022::1
Connection check to gateway:2001:DB8:303:2022::1 - Success
Checking dns server resolution: 2001:DB8:68d:4001::a
Status check to dns server 2001:DB8:68d:4001::a - Success
Network interface - PRIMARY status check successful.

Network interface - SECONDARY
---------------------------------------

Checking connection status for network stack:ipv6
Checking connection to ::1
Connection check to local:::1 - Success
Checking connection to 2001:DB8:303:2021::210
Connection check to ip address:2001:DB8:303:2021::210 - Success
Checking connection to 2001:DB8:303:2021::1
Connection check to gateway:2001:DB8:303:2021::1 - Success
Checking dns server resolution: 2001:DB8:68d:4001::a
Status check to dns server 2001:DB8:68d:4001::a - Success
Network interface - SECONDARY status check successful.
```

The following example shows how to display the network connectivity details of the PRIMARY interface.

```
[spacesadmin@connector ~]$ connectorctl network status -n PRIMARY
Executing command:network
Command execution status:Success
----------------------
=================Network Status=================

Network interface - PRIMARY
---------------------------------------
```

```
Checking connection status for network stack:ipv6
Checking connection to ::1
Connection check to local:::1 - Success
Checking connection to 2001:DB8:303:2022::60
Connection check to ip address:2001:DB8:303:2022::60 - Success
Checking connection to 2001:DB8:303:2022::1
Connection check to gateway:2001:DB8:303:2022::1 - Success
Checking dns server resolution: 2001:DB8:68d:4001::a
Status check to dns server 2001:DB8:68d:4001::a - Success
Network interface - PRIMARY status check successful.
```

The following example shows how to display the network connectivity details of the SECONDARY interface.

```
[spacesadmin@connector ~]$ connectorctl network status -n SECONDARY
Executing command:network
Command execution status:Success
-----------------------
=================Network Status=================

Network interface - SECONDARY
--------------------------------------

Checking connection status for network stack:ipv6
Checking connection to ::1
Connection check to local:::1 - Success
Checking connection to 2001:DB8:303:2021::210
Connection check to ip address:2001:DB8:303:2021::210 - Success
Checking connection to 2001:DB8:303:2021::1
Connection check to gateway:2001:DB8:303:2021::1 - Success
Checking dns server resolution: 2001:DB8:68d:4001::a
Status check to dns server 2001:DB8:68d:4001::a - Success
Network interface - SECONDARY status check successful.
```

Note The above example assumes the following:

- The PRIMARY interface of the connector is on the 2001:DB8:303:2022::0/64 subnet, and is used to communicate with Cisco Spaces.
- The SECONDARY interface of the connector is on the 2001:DB8:303:2021::0/64 subnet, and is used to communicate with all the devices, such as wireless controllers, switches, and APs.

connectorctl network reset

To reset the network configuration of the secondary interface, use the **connectorctl network reset** command.

connectorctl network reset

Command History

Release 3	This command is introduced.

Examples

The following example shows how to reset the network configuration of the secondary interface.

```
[spacesadmin@connector ~]$ connectorctl network reset
Executing command:network
Command execution status:Success
-----------------------
Cleaning all unused connections
Connection 'SECONDARY' (f3f21bf5-f5c6-49cc-8cbd-70c582735466) successfully deleted.
Successfully reset interface:SECONDARY configuration
System reboot will happen in 10 seconds. Do not execute any other command...
```

connectorctl network hostname

To edit the host name of this connector instance, use the **connectorctl network hostname** command.

connectorctl network hostname -n *hostname*

Command History

Release 3	This command is introduced.

Examples

The following is a sample output of the command:

```
[spacesadmin@connector ~]$ connectorctl network hostname -n connector3
Executing command:network
Command execution status:Success
-----------------------
Updated hostname:connector3
```

connectorctl network ipv6

To manage IPv6 routing on a specified interface, use the **connectorctl network ipv6** command.

connectorctl network ipv6 -i *interface-name* { **show** | **enable** | **disable** }

Syntax Description

Keywords and Variables	Description
show	Shows the IPv6 routing status of a specified interface
enable	Enables IPv6 routing on a specified interface
-i *interface-name*	Interface Name
disable	Disables IPv6 routing on a specified interface

Command History

Release 3	This command is introduced.

Usage Guidelines

Note Unless you have enabled IPv6 during the installation of Cisco Spaces: Connector, IPv6 is disabled by defualt.

Examples

The following example shows how to view the IPv6 status of the specified interface (ens32).

```
[spacesadmin@connector ~]$ connectorctl network ipv6 -i ens32 -s
Executing command:network
Command execution status:Success
-----------------------
IPv6 Status:
-----------------------------------------------------
net.ipv6.conf.ens32.disable_ipv6 = 0
```

The following example shows how to disable IPv6 on the specified interface (ens32).

```
[spacesadmin@connector ~]$ connectorctl network ipv6 -i ens32 -d
Executing command:network
Command execution status:Success
-----------------------
Connection successfully activated (D-Bus active path:
/org/freedesktop/NetworkManager/ActiveConnection/5)
IPv6 disabled on interface: ens32
-----------------------------------------------------
net.ipv6.conf.ens32.disable_ipv6 = 1
```

The following example shows how to enable IPv6 on the specified interface (ens32).

```
[spacesadmin@connector ~]$ connectorctl network ipv6 -i ens32 -e
Executing command:network
Command execution status:Success
-----------------------
Connection successfully activated (D-Bus active path:
/org/freedesktop/NetworkManager/ActiveConnection/6)
```

```
IPv6 enabled on interface: ens32
------------------------------------------------------
net.ipv6.conf.ens32.disable_ipv6 = 0
```

IP Route Commands

connectorctl ip-route show

To display the current route configured for this connector instance, use the **connectorctl ip-route show** command. To see only individual interface network details, use the **-n** keyword.

connectorctl ip-route show { **-p** *network-stack* | **-n** *interface* | **-d YES** }

Syntax Description

Keywords and Variables	Description
-p *network-stack*	Accepted values are IPv4 and IPv6.
-d YES	Logs detailed firewall and IP table rules.
-n *interface-name*	Interface name. Accepted values are: • PRIMARY • SECONDARY

Command History

Release 3	This command is introduced.

Examples

The following is a sample output of the command:

```
[spacesadmin@connector ~]$ connectorctl ip-route show -p ipv4 -n SECONDARY

Executing command:ip-route
Command execution status:Success
-----------------------
Route information for connector
10.7.0.0/24 dev ens33 proto kernel scope link src 10.7.0.11 metric 101
10.7.0.0/24 via 10.7.0.1 dev ens33 proto static src 10.7.0.11 metric 101

Rule information for connector
438:    from all to 10.7.0.11 lookup 18
439:    from 10.7.0.11 lookup 18
```

```
[spacesadmin@connector ~]$ connectorctl ip-route show -p ipv4 -n PRIMARY
 Executing command:ip-route
Command execution status:Success
-----------------------
Route information for connector
default via 10.22.244.1 dev ens32 proto static metric 100
10.22.244.0/24 dev ens32 proto kernel scope link src 10.22.244.180 metric 100
```

Note The above example assumes the following:

- The PRIMARY interface of the connector is on the 10.22.x.x subnet, and is used to communicate with Cisco Spaces

  .
- The SECONDARY interface of the connector is on the 7.7.x.x subnet, and is used to communicate with all the devices, such as wireless controllers, switches, and APs.

connectorctl ip-route add

To configure a route for the secondary interface, use the **connectorctl ip-route add** command.

connectorctl ip-route add { **-n** *interface* | **-p** *network-stack* | **-s** *network-subnet* | **-g** *gateway* }

Syntax Description

Keywords and Variables	Description
-n *interface-name*	Interface name. Accepted values are: • PRIMARY • SECONDARY
-p *network-stack*	Accepted values are IPv4 and IPv6.
-s *subnet/ prefix*	Network subnet slash prefix as comma separated list. For example, 10.7.0.11/24.
-g *gateway*	Gateway address or next hop address

Command History

Release 3	This command is introduced.

Examples

The following example shows how to configure an IPv4 route for the secondary interface, on the subnet 10.7.0.11 and prefix 24, and gateway IP address 10.7.0.1.

```
[spacesadmin@connector ~]$ connectorctl  ip-route add -n SECONDARY -p ipv4 -s 10.7.0.11/24
 -g 10.7.0.1

Executing command:ip-route
Command execution status:Success
-----------------------
Adding subnet route:10.7.0.11/24
Successfully added route configuration.
```

Examples

The following example shows how to configure an IPv6 route for the secondary interface, on the subnet 2001:DB8:303:2021::201 and prefix 64, and gateway IP address 2001:DB8:303:2021::1 .

```
connectorctl ip-route add -n SECONDARY -p ipv6 -s 2001:DB8:303:2021::201/64 -g
2001:DB8:303:2021::1


Executing command:ip-route
Command execution status:Success
-----------------------
Adding subnet route:2001:DB8:303:2021::201/64
Successfully added route configuration.
```

connectorctl ip-route delete

To delete the current route configured for the secondary interface, use the **connectorctl ip-route delete** command.

connectorctl ip-route delete { **-p** *network-stack* | **-n** *interface* | **-d YES** }

Syntax Description

Keywords and Variables	Description
-p *network-stack*	Accepted values are IPv4 and IPv6.
-d YES	Logs detailed firewall and IP table rules.
-n *interface-name*	Interface name. Use of the following values: • PRIMARY • SECONDARY

Command History

Release 3	This command is introduced.

Examples

The following example shows how to delete a configured route.

```
[spacesadmin@connector ~]$ connectorctl  ip-route delete -n SECONDARY -p ipv4 -s 10.7.0.0/24

 Executing command:ip-route
Command execution status:Success
-----------------------
Deleting subnet route:10.7.0.0/24 10.7.0.1 src=10.7.0.11
Successfully removed route configuration.
```

Examples

The following example shows how to delete a configured route.

```
[spacesadmin@connector ~]$ connectorctl ip-route add -n SECONDARY -p ipv6 -s
2001:DB8:303:2021::201/64 -g 2001:DB8:303:2021::1
Executing command:ip-route
Command execution status:Success
-----------------------
Adding subnet route:2001:DB8:303:2021::201/64
Successfully added route configuration.
```

Services Commands

connectorctl service restart

This command restarts all the Cisco Spaces: Connector services. To enable debug logs, use the -l keyword is specified.

connectorctl service restart **-s** *service-name* [**-l** *debug-level* [**-d** *debug-period-in-minutes*]]

Syntax Description

Keyword and Variable	Description
-s *service-name*	Configure the service that needs to be restarted.
-l *debug-level*	(Optional) Configure the debug level. Values are DEBUG, INFO, and WARNING. • If *debug-level* is unspecified, the default value is DEBUG. **Note** Running the service at DEBUG log level would significantly impact performance
-d *debug-period-in-minutes*	(Optional) Specify the debug period in minutes. If unspecified, the default value is 10 minutes. • If **-l** is unspecified, service is restarted but debugging is not logged.

Command History

Release 3	This command is introduced.

Examples

You can also restart a specified service. The following is a sample output of the command:

```
$[spacesadmin@connector ~]$ connectorctl service restart -s location -l DEBUG
Executing command:service
Command execution status: Success

Status:Successfully started location
```

connectorctl service status

To display the status of all the services running on the Cisco Spaces: Connector, use the **connectorctl service status** command.

connectorctl service status [**-s** *service-name*]

Syntax Description

Keywords and Variables	Description
-s *service-name*	Displays the status of a specified service only.

Command History

Release 3	This command is introduced.

Examples

You can view the status of all configured services. The following is a sample output of the command:

```
[spacesadmin@ connector ~]$ connectorctl service status
Executing command:service
Command execution status:Success

location(3.0.1.266) status:Up(1 m 34s)
```

You can view the status of a specified service. The following is a sample output of the command:

```
[spacesadmin @ connector ~ ]S connectorctl service status -s location
Executing command:service
Command execution status: Success

location (3.0.1.266) status:Up(5m 50s)
```

connectorctl service stop

To stop the specified service running on the Cisco Spaces: Connector, use the **connectorctl service stop** command. .

connectorctl service stop [**-s** *service-name*]

Syntax Description

Keywords and Variables	Description
-s *service-name*	Stops the specified service.

Command History

Release 3	This command is introduced.

Examples

The following is a sample output of the command:

```
[spacesadmin@connector ~]$ connectorctl service stop -s location
Executing command:service
Command execution status:Success
-----------------------
Status:Successfully stopped location
```

connectorctl service network

To configure the Cisco Spaces: Connector services network, use the **connectorctl service network** command.

connectorctl service network { **-r** | **-i** *ip-address* | **-c** *cidr* | **-s** }

Syntax Description

Keywords and Variables	Description
-r	Resets services network to default.
-i *ip-address*	Base IPv4 address or IPv6 address.
-c *cidr*	Classless Inter-Domain Routing (CIDR) range.
-g	IPv4 or IPv6 gateway address.
-s	Shows current service network configuration.

Command History

Release 3	This command is introduced.

Examples

You can configure the IPv4 address of the network and the CIDR range.The following is a sample output of the command:

```
[spacesadmin@ connector ~ ]$ connectorctl service network -i 172.18.0.0 -c 16 -g 172.18.0.1
Executing command:service
Command execution status:Success

Successfully updated service network to given config
```

You can view the current service network configurations. The following is a sample output of the command:

```
[spacesadmin @ connector ~ ]S connectorctl service network -s

Executing command: service
Command execution status: Success

Base Ip Address: 172.18.0.0
cidr range: 16,
Gateway: 172.18.0.1
```

Examples

You can also configure the IPv6 address of the network and the CIDR range. The following is a sample output of the command:

```
[spacesadmin@ connector ~ ]$ connectorctl service network -i 172.18.0.0 -c 16 -g 172.18.0.1
 -i 2001:db8:1::
-c 64 -g 2001:db8:1::1

Executing command:service
Command execution status:Success

Successfully updated service network to given config
```

You can view the current service network configurations. The following is a sample output of the command:

```
[spacesadmin @ connector ~ ]S connectorctl service network -s

Executing command:service
Command execution status:Success
-----------------------
Base Ip Address: 172.18.0.0
2001:db8:1::
cidr range: 16,
64,
Gateway: 172.18.0.1
2001:db8:1::1
```

System Services Commands

connectorctl systemservice service-manager

To restart or view the status of the Service Manager service, use the **connectorctl systemservice service-manager** command. To enable debug logs, use the **-l** keyword.

connectorctl systemservice service-manager -r [**-l** *debug-level* [**-d** *debug-period-in-minutes*]]

connectorctl systemservice service-manager -s

Syntax Description

Keywords and Variables	Description
-s	Shows the running status of the service manager
-r	Restarts the service manager service.
-l *debug-level*	Restarts the service manager service and configures debugging at the specified *debug-level*. The values are DEBUG, INFO, and WARNING. • If *debug-level* is unspecified, the default value is DEBUG. **Note** Running the service at DEBUG log level could significantly impact performance
-d *debug-period-in-minutes*	(Optional) Specify the debug period in minutes. If unspecified, the default value is ten minutes. • If **-l** is unspecified, the service manager service is restarted but debugging is not logged.

Command History

Release 3	This command is introduced.

Examples

The following example shows how to view the running status of the service manager service.

```
[spacesadmin@connector ~]$ connectorctl systemservice service-manager -s
Executing command:systemservice
Command execution status:Succescs
-----------------------
Service Manager is running with version:3.0.1.41(9m 47s)
```

The following example shows how to restart the service manager service at a specified debug level.

```
[spacesadmin@connector ~]$ connectorctl systemservice service-manager -r [-l DEBUG -d 1]
Executing command:systemservice
Command execution status:Success
-----------------------
Scheduled Service Manager restart. It will take few minutes
```

connectorctl systemservice service-agent

To restart or view the status of the service agent service, use the **connectorctl systemservice service-agent** command. To enable debug logs, use the **-l** keyword.

connectorctl systemservice service-agent -r [**-l** *debug-level* [**-d** *debug-period-in-minutes*]]

connectorctl systemservice service-agent -s

Syntax Description

Keywords and Variables	Description
-s	Shows the running status of the service agent service.
-r	Restarts the service agent service.
-l *debug-level*	Restarts service agent service and configure debugging at the specified *debug-level*. The values are DEBUG, INFO, and WARNING. • If *debug-level* is unspecified, the default value is DEBUG. **Note** Running the service at DEBUG log level would significantly impact performance
-d *debug-period-in-minutes*	(Optional) Debug period in minutes. If unspecified, the default value is 10 minutes. • If **-l** is unspecified, the service agent service is restarted but debugging is not logged.

Command History

Release 3	This command is introduced.

Examples

The following example shows how to see the running status of the service:

```
[spacesadmin@connector ~]$ connectorctl systemservice service-agent -s
Executing command:systemservice
Command execution status:Success
----------------------
==================service-agent status=================
service-agent.service - service agent starting script
   Loaded: loaded (/etc/systemd/system/service-agent.service; enabled; vendor preset:
disabled)
   Active: active (running) since Thu 2022-07-28 12:20:55 PDT; 1 months 9 days ago
 Main PID: 859 (run_service_age)
    Tasks: 7 (limit: 24285)
   Memory: 118.7M
   CGroup: /system.slice/service-agent.service
           ├─859 /bin/bash /opt/dnaspaces-connector/static/service-agent/run_service_agent.sh

             └─902 python3.9 src/application.py

Warning: Journal has been rotated since unit was started. Log output is incomplete or
unavailable.
==========================end============================
```

The following example shows how to restart the service at a specified debug level:

```
[spacesadmin@connector ~]$ connectorctl systemservice service-agent -r [-l DEBUG -d 1]
Executing command:systemservice
Command execution status:Success
-----------------------
Restarted service-agent service
```

connectorctl systemservice docker

To restart or view the status of the **docker** service, use the **connectorctl systemservice docker** command.

connectorctl systemservice docker { **-s** | **-r** }

Syntax Description

Keywords and Variables	Description
-s	Shows the status of the docker service.
-r	Restarts the docker service

Command History

Release 3	This command is introduced.

Examples

The following example shows how to view the status of the docker service.

```
[spacesadmin@connector ~]$ connectorctl systemservice docker -s
Executing command:systemservice
Command execution status:Success
----------------------
=================docker status=================
 docker.service - Docker Application Container Engine
   Loaded: loaded (/usr/lib/systemd/system/docker.service; enabled; vendor preset: disabled)

    Active: active (running) since Thu 2022-07-28 12:21:19 PDT; 1 months 9 days ago
      Docs: https://docs.docker.com
 Main PID: 1180 (dockerd)
    Tasks: 10
    Memory: 186.6M
    CGroup: /system.slice/docker.service
            └─1180 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock

Sep 06 01:36:29 conn3-la61-212-23 dockerd[1180]: time="2022-09-06T01:36:29.281017425-07:00"
 level=info msg="ignoring event"
container=19f30610cb760c8d73abb6d6119a3d9f479334cf1d914242b75cbd2a54590018
module=libcontainerd namespace=moby topic=/tasks/delete type="*events.TaskDelete"
Sep 06 01:36:34 conn3-la61-212-23 dockerd[1180]: time="2022-09-06T01:36:34.512812745-07:00"
 level=info msg="Firewalld: interface docker0 already part of docker zone, returning"
Sep 06 01:36:34 conn3-la61-212-23 dockerd[1180]: time="2022-09-06T01:36:34.669396274-07:00"
 level=info msg="Firewalld: interface br-77efcfe528f1 already part of docker zone, returning"
Sep 06 01:36:34 conn3-la61-212-23 dockerd[1180]: time="2022-09-06T01:36:34.808767362-07:00"
 level=info msg="Firewalld: interface docker0 already part of docker zone, returning"
Sep 06 01:36:34 conn3-la61-212-23 dockerd[1180]: time="2022-09-06T01:36:34.840664442-07:00"
 level=info msg="Firewalld: interface docker0 already part of docker zone, returning"
Sep 06 01:47:06 conn3-la61-212-23 dockerd[1180]: time="2022-09-06T01:47:06.126501950-07:00"
 level=info msg="ignoring event"
container=7bbcc6bc2cb33623f05d57768a4a1bba08b40cad282eddbeb53c8c6dec6157a9
module=libcontainerd namespace=moby topic=/tasks/delete type="*events.TaskDelete"
Sep 06 01:47:11 conn3-la61-212-23 dockerd[1180]: time="2022-09-06T01:47:11.264882240-07:00"
 level=info msg="Firewalld: interface docker0 already part of docker zone, returning"
Sep 06 01:47:11 conn3-la61-212-23 dockerd[1180]: time="2022-09-06T01:47:11.424043921-07:00"
 level=info msg="Firewalld: interface br-77efcfe528f1 already part of docker zone, returning"
Sep 06 01:47:11 conn3-la61-212-23 dockerd[1180]: time="2022-09-06T01:47:11.562900640-07:00"
 level=info msg="Firewalld: interface docker0 already part of docker zone, returning"
Sep 06 01:47:11 conn3-la61-212-23 dockerd[1180]: time="2022-09-06T01:47:11.596127048-07:00"
 level=info msg="Firewalld: interface docker0 already part of docker zone, returning"
========================end============================
```

The following example shows how to restart the docker service.

```
[spacesadmin@connector ~]$ connectorctl systemservice docker -r
Executing command:systemservice
Command execution status:Success
----------------------
Restarted docker service
```

User Authentication Commands

connectorctl userauth lock

To lock out a **spacesadmin** user from the GUI after a specific interval or incorrect password login attempts, use the **connectorctl userauth lock** command.

connectorctl userauth lock { **-d** *deny-attempt-count* | **-i** *interval* | **-s** }

Syntax Description

Keywords and Variables	Description
-d *deny-attempt-count*	Number of incorrect login attempts.
-i *interval*	Lock out the interval.
-s	Shows current configuration.

Command History

Release 3	This command is introduced.

Examples

The following example shows how to lock out the **spacesadmin** user from the GUI for a period of 4 minutes after 2 unsuccessful login attempts with an incorrect password:

```
[spacesadmin@ connector ~]$ connectorctl userauth lock -d 2 -i 4
Executing command:userauth
Command execution status:Success

Successfully updated user lock profile
```

The following example shows how to view the current lockout configurations:

```
[spacesadmin @ connector ~ ]S connectorctl userauth lock - s
Executing command:userauth
Command execution status:Success

Deny attempt count:3
Lockout interval:1
```

connectorctl userauth password

To configure the strength of the password, set an expiry period and minimum length of the password, use the **connectorctl userauth password** command.

connectorctl userauth password { **-l** *password-length* | **-p { yes | no }** | **-r { yes | no }** | **-e { yes | no }** | **-u** *password-reuse* | **-s** }

Syntax Description

Keywords and Variables	Description
-l *password-length*	Minimum password length. Default value is eight.
-p { yes | no }	Enables a strong password. Default value is no.
-r { yes | no }	Rejects a weak password. Default value is yes.
-e { yes | no }	Password expires after 60 days. Default value is no.
-u *password-reuse*	Number of previous passwords that can be reused. Default value is zero.
-s	Shows current configuration

Command History

Release 3	This command is introduced.

Examples

The following example shows how to define the minimum length of the password as 9, configure a flag that requires a strong password, configure a flag that rejects a weak password, set the password to expire after 60 days, and configure that one previous password cannot be reconfigured again.

```
[spacesadmin@connector ~]$ connectorctl userauth password -l 9 -p yes -r yes -e yes -u 1
Executing command:userauth
Command execution status:Success
-----------------------
Updated password policy
User password expiry set to 60 days
Successfully updated user password profile
```

The following example shows how to view the password limitations currently configured.

```
[spacesadmin@connector ~]$ connectorctl userauth password -s
Executing command:userauth
Command execution status:Success
-----------------------
Password length: 8
Enable strong password: no
Reject weak password: yes
Expire password after 60 days: no
Number of previous passwords which cannot be reused: 0
```

connectorctl userauth reset

To reset the user password and lock configuration to system default, use the **connectorctl userauth reset** command.

connectorctl userauth reset

Syntax Description

This command has no keywords or arguments.

Command History

Release 3	This command is introduced.

Examples

```
[spacesadmin@connector ~]$ connectorctl userauth reset
Executing command:userauth
Command execution status:Success
----------------------
User auth profile reset to system default
```

Certificate Commands

connectorctl cert createcsr

To create a connector Certificate Signing Request using the parameters you provide, use the **connectorctl cert createcsr** command.

connectorctl cert createcsr -s *san* **-c** *country* **-t** *state* **-l** *locality* **-o** *organization* **-u** *organizationalunit* **-n** *commonname* **-e** *email*

Syntax Description

Keywords and Variables	Description
-s *san*	Storage Area Network (SAN) name.
-c *country*	Country name.
-t *state*	State name.
-l *locality*	Locality name.
-o *organization*	Organization name.
-u *organizationalunit*	Organizational unit name.
-n *commonname*	Common name.
-e *email*	Email ID.

Command History

Release 3	This command is introduced.

connectorctl cert generate

To regenerate a new connector self-signed certificate, use the **connectorctl cert generate** command. To view this certificate, use the **connectorctl cert show** command.

connectorctl cert generate

Syntax Description

This command has no keywords or arguments.

Command History

Release 3	This command is introduced.

connectorctl cert import

To import a signed certificate from the specified path to the accurate location on the connector and ensure the security of the connection with the connector, use the **connectorctl cert import** command.

connectorctl cert import -p *certificate-path*

Syntax Description

Keywords and Variables	Description
-p *certificate-path*	Path from which the certificate is to imported.

Command History

Release 3	This command is introduced.

Examples

The following is a sample output of the command:

```
[spacesadmin@connector ~]$ connectorctl cert import -p
```

connectorctl cert show

To display the deployed certificate details, use the **connectorctl cert show** command.

connectorctl cert show

Syntax Description

This command has no keywords or arguments.

Command History

Release 3	This command is introduced.

Examples

The following is a sample output of the command:

```
[spacesadmin @ connector ~ ]S connectorctl cert show
Executing command:cert
Command execution status:Success
----------------------
================================================================================
Certificate not found.
================================================================================
==========
```

connectorctl cert validate

To validate certificates, use the **connectorctl cert validate** command.

After validating the certificate, you can upload the certificates to the connector using the **connectorctl cert updateca-bundle** command.

connectorctl cert validate -c *ca_certificate* **-s** *path_server_certificate*

Syntax Description

Keywords and Descriptions	Description
-h	Displays help related to this command.
-c *ca_certificate*	Signed and validated CA certificate.
-s *path_server_certificate*	Signed and validated server certificate.

Command History

Release 3	This command is introduced.

Usage Guidelines

First, copy the certificates to connector.

```
scp proxy-ca-bundle.pem spacesadmin@[connector-ip]:/home/spacesadmin/
scp proxy-server-cert.pem spacesadmin@[connector-ip]:/home/spacesadmin/
```

Examples

Validate the copied certificate. The following is a sample output of the command:

```
[spacesadmin@connector ~]$ connectorctl cert validate -c /home/spacesadmin/proxy-ca-bundle.pem
 -s /home/spacesadmin/proxy-server-cert.pemExecuting command:certCommand execution
status:Success----------------------/home/spacesadmin/proxy-ca-bundle.pem and
/home/spacesadmin/proxy-server-cert.pem exists/home/spacesadmin/proxy-server-cert.pem:
OKValidation of certificate is successful
```

connectorctl cert updateca-bundle

To import a Certification Authority (CA) chain to the the connector's CA trust bundle, use the **connectorctl cert updateca-bundle** command.

connectorctl cert updateca-bundle -c *ca_certificate_chain* **-s** *server_certificate*

Syntax Description

Keywords and Variables	Description
-c *ca_certificate*	Signed and validated CA certificate.
-s *server_certificate*	Signed and validated server certificate.

Command History

Release 3	This command is introduced.

Usage Guidelines

First, copy the certificates to connector.

```
scp proxy-ca-bundle.pem spacesadmin@[connector-ip]:/home/spacesadmin/
scp proxy-server-cert.pem spacesadmin@[connector-ip]:/home/spacesadmin/
```

Examples

Import the copied certificates. The following is a sample output of the command:

```
[spacesadmin@connector ~]$ connectorctl cert updateca-bundle -c
/home/spacesadmin/proxy-ca-bundle.pem -s /home/spacesadmin/proxy-server-cert.pem
Executing command:cert
Command execution status:Success
-----------------------
/home/spacesadmin/proxy-ca-bundle.pem and /home/spacesadmin/proxy-server-cert.pem exist
/home/spacesadmin/proxy-server-cert.pem: OK
CA trust bundle updated successfully
System reboot will happen in 10 seconds. Do not execute any other command...
```

connectorctl cert proxycert-validate

To validate proxy certification authority (CA) bundle, use the **connectorctl cert proxycert-validate** command.

To validate certificates before uploading them to connector, use the **connectorctl cert proxycert-updateca-bundle** command.

connectorctl cert proxycert-validate -c *proxy-ca-cert-chain* **-s** *proxy_server_certificate*

Syntax Description

Keywords and Descriptions	Description
-h	Displays help related to this command.
-c*proxy-ca-certificate-chain*	Signed and validated proxy CA certificate.
-s *proxy-server-certificate*	Signed and validated proxy server certificate.

Command History

Release 3	This command is introduced.

Usage Guidelines

First, copy the certificates to connector.

```
scp proxy-ca-bundle.pem spacesadmin@[connector-ip]:/home/spacesadmin/
scp proxy-server-cert.pem spacesadmin@[connector-ip]:/home/spacesadmin/
```

Examples

Validate the copied certificate. The following is a sample output of the command:

```
[spacesadmin@connector ~]$ connectorctl cert validate -c /home/spacesadmin/proxy-ca-bundle.pem
 -s /home/spacesadmin/proxy-server-cert.pem
Executing command:certCommand execution status:Success
----------------------
/home/spacesadmin/proxy-ca-bundle.pem and /home/spacesadmin/proxy-server-cert.pem
exists/home/spacesadmin/proxy-server-cert.pem: OK
Validation of certificate is successful
```

connectorctl cert proxycert-updateca-bundle

This command imports a proxy Certification Authority (CA) chain to the the connector's CA trust bundle.

connectorctl cert proxycert-updateca-bundle -c *proxy-ca-certificate-chain* **-s** *proxy-server-certificate*

Syntax Description

Keywords and Variables	Description
-c *proxy-ca-certificate-chain*	Provides the signed and validated proxy CA certificate.
-s *proxy-server-certificate*	Provides the signed and validated proxy server certificate.

Command History

Release 3	This command is introduced.

Usage Guidelines

First, copy the certificates to connector.

```
scp proxy-ca-bundle.pem spacesadmin@[connector-ip]:/home/spacesadmin/
scp proxy-server-cert.pem spacesadmin@[connector-ip]:/home/spacesadmin/
```

Examples

Import the copied certificates. The following is a sample output of the command:

```
[spacesadmin@connector ~]$ connectorctl cert updateca-bundle -c
/home/spacesadmin/proxy-ca-bundle.pem -s /home/spacesadmin/proxy-server-cert.pem
Executing command:cert
Command execution status:Success
-----------------------
/home/spacesadmin/proxy-ca-bundle.pem and /home/spacesadmin/proxy-server-cert.pem exist
/home/spacesadmin/proxy-server-cert.pem: OK
CA trust bundle updated successfully
System reboot will happen in 10 seconds. Do not execute any other command...
```

RSyslog Commands

connectorctl rsyslog update

To update the Rsyslog service, use the **connectorctl rsyslog update** command.

connectorctl rsyslog update *protocol-name ip-address port-number SAN path-ca-cert*

Syntax Description

Keywords and Variables	Description
protocol-name	Updates RSyslog protocols. Possible values are UDP, TCP, or TLS.
ip-address	Updates IP address configured for the RSyslog service protocols.
port-number	Updates port number configured for the RSyslog service protocols.
SAN	Updates Storage Area Network (SAN) number configured for the RSyslog service protocols. (For Transport Layer Security [TLS] only)
path-ca-cert	Signed and validated CA certificate. (For TLS only)

Command History

Release 3	This command is introduced.

connectorctl rsyslog show

To display details of the the Rsyslog service, use the **connectorctl rsyslog show** command.

connectorctl rsyslog show

Syntax Description

This command has no keywords or arguments.

Command History

Release 3	This command is introduced.

connectorctl rsyslog restart

To restart the Rsyslog service, use the **connectorctl rsyslog restart** command.

connectorctl rsyslog restart

Syntax Description

This command has no keywords or arguments.

Command History

Release 3	This command is introduced.

connectorctl rsyslog reset

To reset the Rsyslog service, use the **connectorctl rsyslog reset** command.

connectorctl rsyslog reset

Syntax Description

This command has no keywords or arguments.

Command History

Release 3	This command is introduced.

High Availability

connectorctl ha failover

This command initiates failover to a backup connector instance.

connectorctl ha failover

Syntax Description

This command has no keywords or arguments.

Examples

The following example shows how to initiate failover to a backup connector instance.

```
spacesadmin@connector ~]$ connectorctl ha failover

Executing command:ha
Command execution status:Success
-----------------------
HA failover triggered. This process will take around 30 seconds.
```

connectorctl ha show

To show the high availability configuration, use the **connectorctl ha show** command.

connectorctl ha show

Examples

The following example shows how to SSH to a connector in VIP-paired mode before failover.

```
[spacesadmin@connector ~]$ connectorctl ha show
Executing command:ha
Command execution status:Success
-----------------------
mode: VIP Paired
ha_state: ACTIVE
vip: 10.89.45.94
peer_ip: 10.89.45.92
peer_instance_id: 005056a754c8
instance_channel_status: UP
```

The following example shows how to SSH to the active connector instance after failover.

```
[spacesadmin@connector ~]$ connectorctl ha show
Executing command:ha
Command execution status:Success
-----------------------
HA failover triggered. This process will take around 30 seconds.
[spacesadmin@conn-sec ~]$

[spacesadmin@conn-pri ~]$ connectorctl ha show
Executing command:ha
Command execution status:Success
-----------------------
mode: VIP Paired
ha_state: ACTIVE
vip: 10.89.45.94
peer_ip: 10.89.45.93
peer_instance_id: 005056a7affa
instance_channel_status: UP
```

The following is a sample output of the command on the backup connector of the VIP pair.

```
[spacesadmin@connector ~]$ connectorctl ha show
Executing command:ha
Command execution status:Success
-----------------------
mode: VIP Paired
ha_state: BACKUP
vip: 10.89.45.94
peer_ip: 10.89.45.92
peer_instance_id: 005056a754c8
instance_channel_status: UP
```

connectorctl ha restart

To restart the Keepalived services on a connector, use the **connectorctl ha restart** command.

Note Keepalived is a service that establishes the High Availability, and maintains the state of High Availability.

connectorctl ha restart

Syntax Description

This command has no keywords or arguments.

Examples

The following is a sample output of the command:

```
[spacesadmin@connector ~]$ connectorctl ha restart
Executing command:ha
Command execution status:Success
-----------------------
Keepalived service restarted successfully.
● keepalived.service - LVS and VRRP High Availability Monitor
   Loaded: loaded (/usr/lib/systemd/system/keepalived.service; enabled; vendor preset:
disabled)
   Active: active (running) since Tue 2023-04-25 14:21:10 PDT; 2s ago
```

Note Executing this command on an SSH session with a connector configured in VIP mode terminates the SSH session.

connectorctl ha history

To show the history of high availability status, use the **connectorctl ha history** command.

connectorctl ha history

Syntax Description

This command has no keywords or arguments.

Examples

The following is a sample output of the command:

```
[spacesadmin@connector ~]$ connectorctl ha history
Executing command:ha
Command execution status:Success
----------------------
Recent HA states and corresponding timestamps are displayed below for instance with IP
address: 172.19.28.90
Current state of instance: BACKUP
Apr 24 13:35:10 172 Keepalived_vrrp[1239]: (VRRP1) Entering FAULT STATE
Apr 24 13:35:20 172 Keepalived_vrrp[1239]: (VRRP1) Entering BACKUP STATE
Apr 24 17:42:51 172 Keepalived_vrrp[139964]: (VRRP1) Entering BACKUP STATE
Apr 24 17:43:21 172 Keepalived_vrrp[139964]: (VRRP1) Entering ACTIVE STATE
Apr 24 19:42:31 172 Keepalived_vrrp[176498]: (VRRP1) Entering BACKUP STATE
```

AAA Commands

connectorctl aaa config

To help in configuring the Authentication, Authorization, and Accounting (AAA) server, use the **connectorctl aaa config** command.

connectorctl aaa config *host-ip port secret-key*

Syntax Description

Keywords and Variables	Description
host-ip	IP address of the AAA server.
port	Port used to connect to the AAA server. Default value is 1812.
secret-key	Shared secret key used to connect to the AAA server

Command History

Release 3	This command is introduced.

Examples

The following example shows how to configure the network with an IP address and secret key.

```
[spacesadmin@connctor ~]$ connectorctl aaa config 10.XX.XX.XX XXXX testing123
Executing command:aaa
Command execution status:Success
-----------------------
Connection to AAA Server Successful. AAA Settings are correct. Please wait for 2 minutes
to login to the UI
```

connectorctl aaa disable

To disable the Authentication, Authorization, and Accounting (AAA) configurations on Cisco Spaces: Connector, use the **connectorctl aaa disable** command.

connectorctl aaa disable

Syntax Description

This command has no keywords or arguments.

Command History

Release 3	This command is introduced.

Examples

The following example shows how to disables AAA configurations.

```
$[spacesadmin@connector ~]$ connectorctl aaa disable
Executing command:aaa
Command execution status:Success
----------------------
================================================================================
AAA server is disabled successfully
================================================================================
```

connectorctl aaa show

To show the Authentication, Authorization, and Accounting (AAA) server configuration made on Cisco Spaces: Connector, use the **connectorctl aaa show** command..

connectorctl aaa show

Syntax Description

This command has no keywords or arguments.

Command History

Release 3	This command is introduced.

Examples

The following example shows how to display AAA configurations.

```
[spacesadmin@connector ~]$ connectorctl aaa show
Executing command:aaa
Command execution status:Success
-----------------------
================================================================================
AAA Server is Enabled
================================================================================
AAA Server IP : 10.XX.XX.XX
================================================================================
AAA Server PORT : XXXX
================================================================================
Shared Secret : **<<masked>>**
================================================================================
```

connectorctl aaa ipsec-config

To configure the IP Security tunnel established from the Cisco Spaces: Connector to the existing Authentication, Authorization, and Accounting (AAA) server, use the **connectorctl aaa ipsec-config** command.

connectorctl aaa ipsec-config *dns-name-of-aaa-server* **authtype** *authentication-type certfile-for-public-key* **autogen** *autogen-methods psk-from-aaa-server*

Syntax Description

Keywords and Variables	Description
dns-name-of-aaa-server	Domain Name Server (DNS) name of the AAA server.
authtype *authentication-type*	Chooses between IPSec Authentication, namely **pubkey** or **PSK**.
certfile-for-public-key	AAA server's CA certificate file
autogen *autogen-methods*	Chooses between two types of autogen methods: • **a**: Choose to autogenerate the PSK. • **p**: Choose to provide the PSK configured on the AAA server.
psk-from-aaa-server	PSK value existing on the AAA server.

Command History

Release 3	This command is introduced.

connectorctl aaa ipsec-autogen-psk

To help activate IP Security tunnel configured on Cisco Spaces: Connector to the existing Authentication, Authorization, and Accounting (AAA) server after autogenerating preshared keys (PSK) on the AAA server, use the **connectorctl aaa ipsec-autogen-psk** command.

connectorctl aaa ipsec-autogen-psk

Syntax Description

This command has no keywords or arguments.

Command History

Release 3	This command is introduced.

Troubleshooting Commands

connectorctl troubleshoot connectivity

This command troubleshoots the connection between Cisco Spaces: Connector and Cisco Spaces. Check the connection before and after the installation of the token on connector.

connectorctl troubleshoot connectivity -r *region* **-e** *environment* [**-p** *proxy-flag* [**-v** *proxy-url*]]

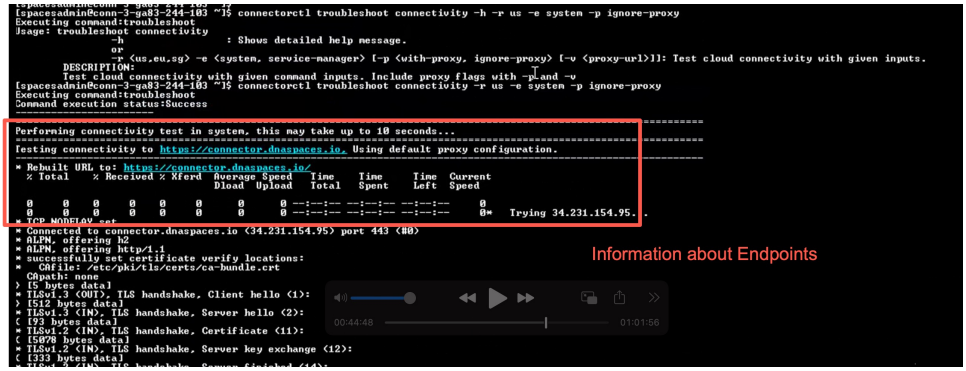
Keywords and Variables	Description
-r *region*	Configures the region. Choose from the following values: • us: United States of America • eu: Europe • sg: Singapore
-e *environment*	Configures the environment in which you want troubleshooting. Choose from the following values: • system: Checks connectivity to Cisco Spaces. • ignore-proxy: Checks if services are able to reach Cisco Spaces
-p *proxy-flag*	Configures the proxy flag. Choose from the following values: • with-proxy: Specify this option when you do not have a proxy on connector • ignore-proxy: Specify this option when you have a proxy configured on connector.
-v *proxy-url*	If you have a proxy configured on connector, use this keyword to specify the proxy IP address.

Command History

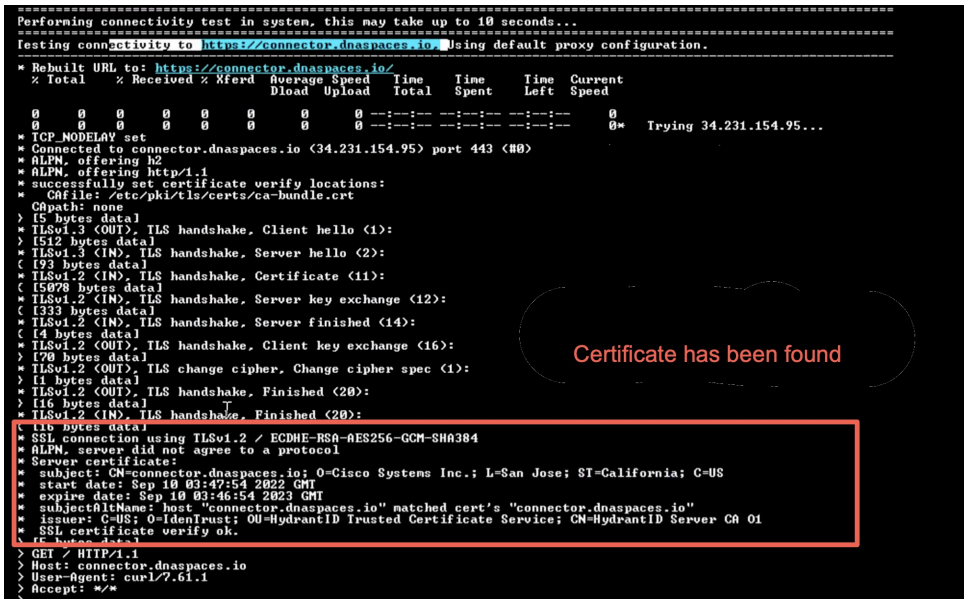
Release 3	This command is introduced.

Examples

Information About Endpoints Found



Certificate Found



connectorctl troubleshoot bandwidth

To check the bandwidth of the connection between Cisco Spaces: Connector and Cisco Spaces, use the **connectorctl troubleshoot bandwidth** command.

connectorctl troubleshoot bandwidth { **-u** | **-d** }

Keywords and Variables	Description
-u	Checks the upload speed between Cisco Spaces: Connector and Cisco Spaces
-d	Checks the download speed between Cisco Spaces: Connector and Cisco Spaces

Command History

Release 3	This command is introduced.

Examples

The following example shows how to view the upload speed between Cisco Spaces: Connector and Cisco Spaces.

```
[spacesadmin@connector ~]$ connectorctl  troubleshoot bandwidth -u
Executing command:troubleshoot
Command execution status:Success
-----------------------
Spaces Cloud Endpoint: https://connector.qa-dnaspaces.io
Test file size: 10MB
Upload Speed:0.6 Mbps.
```

The following example shows how to view the download speed between Cisco Spaces: Connector and Cisco Spaces.

```
[spacesadmin@connector ~]$ connectorctl  troubleshoot bandwidth -d
Executing command:troubleshoot
Command execution status:Success
-----------------------
Spaces Cloud Endpoint: https://connector.qa-dnaspaces.io
Test file size: 10MB
Download Speed:0.6 Mbps.
```

Service Endpoint Commands

Service endpoints are external endpoints defined on the connector,such as GUI, SSH access, Location, fast path, and so on. These endpoints are used by customer and different devices (such as wireless controllers) to connect to the connector and enable network data traffic to Cisco Spaces.

connectorctl service-endpoint show

This command shows the permissions granted to a services-endpoint on an interface. For example, a service can be allowed and/or denied on the primary or the secondary interface.

connectorctl network show [**-n** *interface-name*]

Syntax Description

Keywords and Variables	Description
-n *interface-name*	Specifies interface name. Accepted values are: • PRIMARY • SECONDARY

Command History

Release 3	This command is introduced.

Examples

The following example shows how to view the permissions configured for service-endpoints on interfaces.

```
[spacesadmin@connector ~]$ connectorctl service-endpoint show
Executing command:service-endpoint
Command execution status:Success
-----------------------
Interface: PRIMARY
---------------------------------
Allowed service endpoint: LOCATION_FASTPATH,LOCATION_TDL,SSH,WEB_UI
Denied service endpoint:

Interface: SECONDARY
---------------------------------
Allowed service endpoint: WEB_UI
Denied service endpoint: LOCATION_FASTPATH,LOCATION_TDL,SSH
```

Note LOCATION_FASTPATH, LOCATION_TDL are allowed on the secondary interface by default.

connectorctl service-endpoint config

This command helps to configure services-endpoints on the primary or secondary interface. We can allow and deny each service on either the primary or the secondary interface.

connectorctl service-endpoint config { **-s** *service-endpoint* | **-a** *allowed-interface-names* | **-d** *denied interface-names* }

Syntax Description

Keywords and Variables	Description
-s *service-endpoint*	Specifies a list of comma separated service-endpoints. You can view these enpoints using the connectorctl service-endpoint show
-a *allowed-interface-names*	Specifies a list of comma separated network interfaces where service-endpoints are allowed.
-d *denied interface-names*	Specifies a list of comma separated network interfaces where service-endpoints are denied.

Command History

Release 3	This command is introduced.

Examples

The following example shows how to configure a service endpoint named WEB_UI, and allows WEB_UI on the SECONDARY interface, and denies WEB_UI on the PRIMARY interface.

```
[spacesadmin@connector ~]$ connectorctl  service-endpoint config -s WEB_UI -a SECONDARY -d
 PRIMARY
Executing command:service-endpoint
Command execution status:Success
-----------------------
Successfully updated allow and deny list for given endpoints.
System reboot will happen in 10 seconds. Do not execute any other command...
```

System Upgrade Commands

connectorctl systemupgrade list

This command shows you if there are any upgrades available for the currently installed version of the connector.

connectorctl systemupgrade list

Syntax Description

This command has no keywords or arguments.

Command History

Release 3	This command is introduced.

Examples

The following example shows how to check for available upgrades and see a list of features and enhancements included in the upgrade.

```
[spacesadmin@connector ~]$ connectorctl systemupgrade list
Executing command:systemupgrade
Command execution status:Success
-----------------------
Package:connector3-p84-jan2023-upgrade2
Size:2.3GB
-------------------------------
Summary:
This upgrade includes improvements around network troubleshooting, proxy ca bundle uploads,
 security updates
-------------------------------
Details:
Upgrade includes
1. Connector Dashboard with network troubleshooting support
2. Connector network troubleshooting cli
3. Enhanced proxy ca bundle upload commands
4. Inline system upgrades from command line
5. Security Updates
-------------------------------
Important Notes:
NA
-------------------------------
```

connectorctl systemupgrade install

This command installs any upgrades available for the currently installed version of connector.

connectorctl systemupgrade install

Syntax Description

This command has no keywords or arguments.

Command History

Release 3	This command is introduced.

connectorctl systemupgrade status

This command shows you the status of an ongoing or queued connector upgrade.

connectorctl systemupgrade status

Syntax Description

This command has no keywords or arguments.

Command History

Release 3	This command is introduced.

Note Ensure that Service manager service is upgraded to the latest from the Cisco Spaces dashboard before doing this CLI upgrade.

Examples

The following example shows how to view the status of an ongoing or queued connector upgrade:

```
[spacesadmin@connector ~]S connectorctl systemupgrade status
Executing command:systemupgrade
Command execution status :Success
Successfully upgraded system to package: connector3-p$1-dec2822-upgrade2 at :Jan-86-2023
00:00:47
```

MAC Debug Commands

connectorctl -s local-firehose macdebug viewdebuglogs

This command displays the mac debug logs on the Cisco Spaces: Connector configured using the **connectorctl -s local-firehose macdebug enable** command

connectorctl -s local-firehose macdebug viewdebuglogs -m *macaddress*

Syntax Description

Keywords and Variables	Description
-m *macaddress*	MAC address of the Radio Frequency Identification (RFID) tag or the Bluetooth Low Energy (BLE) tag that should be debugged.

Examples

The following is a sample output of the command:

```
[spacesadmin@connector ~]$ connectorctl -s local-firehose macdebug viewdebuglogs -m
00:0c:cc:45:f0:b3

Executing command:macdebug
Command execution status:Success
-----------------------
2023-10-05 02:42:03,434 [pool-21-thread-1] INFO  com.cisco.dnaspaces.firehose.Firehose -
MAC_DEBUG: 00:0c:cc:45:f0:b3, Incoming data from Connector, type: TAG_RSSI, message: tenantId:
 "11564"
macAddress: "00:0c:cc:45:f0:b3"
controllerIpAddress: "10.22.244.173"
messageId: 15
measurementNotification {
  tenantId: "11564"
  tenantId: "11564"
  macAddress: "00:0c:cc:45:f0:b3"
  controllerIpAddress: "10.22.244.173"
  deviceCategory {
    deviceClass: TAGS_2
  }
  transmitPower {
    value: 19
  }
  apRssiMeasurements {
    entries {
      apMacAddress: "68:7d:b4:5f:26:c0"
      rssi: -42
      timestamp: 50185
    }
    entries {
      apMacAddress: "ec:f4:0c:0e:9e:e0"
      rssi: -47
      timestamp: 54185
    }
    entries {
      apMacAddress: "10:f9:20:fe:5e:a0"
      rssi: -50
      timestamp: 50185
    }
    entries {
      apMacAddress: "0c:d0:f8:95:f3:80"
      rssi: -51
      timestamp: 49185
    }
```

```
    maxEntry {
      apMacAddress: "68:7d:b4:5f:26:c0"
      rssi: -42
      timestamp: 50185
    }
  }
  ccxTagPayloadList {
    timestamp: 49185
    sequenceNumber: 2426
    data:
"\000\023\v\006\002\000\002\0003\002\a\n\000f\000\000\001\275\003\005\001A\302@\000\004\a\000\f\314\000\000\017\000"

  }
  serviceDescriptor {
    serviceId: RSSI
    serviceMask: 2
  }
}
sourceTimestamp: 1696473723296
```

Command History

Release 3	This command is introduced.

Related Topics

connectorctl -s local-firehose macdebug enable, on page 91
connectorctl -s local-firehose macdebug disable, on page 90

connectorctl -s local-firehose macdebug disable

This command disables the debug mode that you enabled earlier for a MAC address using the **connectorctl local-firehose macdebug enable** command.

connectorctl -s local-firehose macdebug disable -m *macaddress*

Syntax Description

Keywords and Variables	Description
-m *macaddress*	MAC address of the Radio Frequency Identification (RFID) tag or the Bluetooth Low Energy (BLE) tag that you want to debug.

Examples

The following is a sample output of the command:

```
[spacesadmin@connector ~]$ connectorctl -s local-firehose macdebug disable -m
00:0c:cc:45:f0:b3

Executing command:macdebug
Command execution status:Success
-----------------------
debug level cleared successfully for mac address: 00:0c:cc:45:f0:b3
```

Command History

Release 3	This command is introduced.

Related Topics

connectorctl -s local-firehose macdebug enable, on page 91
connectorctl -s local-firehose macdebug viewdebuglogs, on page 88

connectorctl -s local-firehose macdebug enable

This command enables debug mode for a particular MAC address. You can then view the debug logs generated for the MAC address using the **connectorctl -s local-firehose macdebug viewdebuglogs** command.

connectorctl -s local-firehose macdebug enable -m *macaddress* **-l** *level* **-d** *duration*

Syntax Description

Keywords and Variables	Description
-m*macaddress*	MAC address of the Radio Frequency Identification (RFID) tag or the Bluetooth Low Energy (BLE) tag that you want to debug.
-l*level*	Debug level. Values can be any of the following. • MESSAGE: Prints debug messages in human-readable format. • BYTE: Prints debug messages in byte code.
-d*duration*	Debug time in minutes.

Examples

The following is a sample output of the command:

```
[spacesadmin@connector ~]$ connectorctl -s local-firehose macdebug enable -m 00:0c:cc:45:f0:b3
 -l MESSAGE -d 15
Executing command:macdebug
Command execution status:Success
-----------------------
debug level set successfully for mac address: 00:0c:cc:45:f0:b3, level: MESSAGE, duration:
 15
```

Command History

Release 3	This command is introduced.

Related Topics

connectorctl -s local-firehose macdebug viewdebuglogs, on page 88
connectorctl -s local-firehose macdebug disable, on page 90

connectorctl -s location macdebug viewdebuglogs

This command displays the mac debug logs for the location service running on the Cisco Spaces: Connector configured using the **connectorctl -s location macdebug enable** command

connectorctl -s location macdebug viewdebuglogs -m *macaddress*

Syntax Description

Keywords and Variables	Description
-m *macaddress*	MAC address of the Radio Frequency Identification (RFID) tag or the Bluetooth Low Energy (BLE) tag that should be debugged.

Examples

The following is a sample output of the command::

```
[spacesadmin@connector ~]$ connectorctl -s location macdebug viewdebuglogs -m
34:e1:2d:23:0a:7d
Executing command:macdebug
Command execution status:Success
------------------------
2023-12-06 07:28:35,801 [qtp2036843608-15] INFO  com.cisco.cmx.auth.AuthFilter - Successfully
 authorized request
2023-12-06 07:28:36,516 [nioEventLoopGroup-8-1] INFO
com.cisco.cmx.nmsp.protomapping.MappingEngine -
{"tenantId":"10137","macAddress":"34:e1:2d:23:0a:7d","controllerIpAddress":"10.22.243.31","messageId":15,"measurementNotification":
{"tenantId":"10137","macAddress":"34:e1:2d:23:0a:7d","controllerIpAddress":"10.22.243.31","timestamp":"0","deviceCategory":
{"deviceClass":"STATIONS","blockSize":0},"apRssiMeasurements":
{"entries":[{"apMacAddress":"04:eb:40:9f:b0:20","ifSlotId":1,"bandId":1,"antennaId":1,"rssi":-87,"timestamp":1355},
{"apMacAddress":"04:eb:40:9f:b0:20","ifSlotId":1,"bandId":1,"antennaId":0,"rssi":-87,"timestamp":1355},
{"apMacAddress":"04:eb:40:9f:ad:80","ifSlotId":1,"bandId":1,"antennaId":1,"rssi":-86,"timestamp":1346},
{"apMacAddress":"04:eb:40:9f:ad:80","ifSlotId":1,"bandId":1,"antennaId":0,"rssi":-86,"timestamp":1346},
{"apMacAddress":"04:eb:40:9f:ab:20","ifSlotId":1,"bandId":1,"antennaId":1,"rssi":-86,"timestamp":1423},
{"apMacAddress":"04:eb:40:9f:ab:20","ifSlotId":1,"bandId":1,"antennaId":0,"rssi":-86,"timestamp":1423},
{"apMacAddress":"04:eb:40:9f:a7:e0","ifSlotId":1,"bandId":1,"antennaId":1,"rssi":-82,"timestamp":1456},
{"apMacAddress":"04:eb:40:9f:a7:e0","ifSlotId":1,"bandId":1,"antennaId":0,"rssi":-82,"timestamp":1456},
{"apMacAddress":"04:eb:40:9f:ad:c0","ifSlotId":1,"bandId":1,"antennaId":1,"rssi":-80,"timestamp":1348},
{"apMacAddress":"04:eb:40:9f:ad:c0","ifSlotId":1,"bandId":1,"antennaId":0,"rssi":-80,"timestamp":1348}],"maxEntry":
{"apMacAddress":"04:eb:40:9f:ad:c0","ifSlotId":1,"bandId":1,"antennaId":1,"rssi":-80,"timestamp":1348}},"txTagPayloadList":[],"directedStatsList":[],"serviceDescriptor":
{"serviceId":"RSSI","serviceMask":1},"isMacHashed":false},"sourceTimestamp":"1701847716265","ingestTimestamp":"0"}
```

Command History

Release 3	This command is introduced.

Related Topics

connectorctl -s location macdebug enable, on page 94
connectorctl -s location macdebug disable, on page 93

connectorctl -s location macdebug disable

This command disables the debug mode for location service that you enabled earlier for a MAC address using the **connectorctl location macdebug enable** command.

connectorctl -s location macdebug disable -m *macaddress*

Syntax Description

Keywords and Variables	Description
-m *macaddress*	MAC address of the Radio Frequency Identification (RFID) tag or the Bluetooth Low Energy (BLE) tag that you want to debug.

Examples

The following is a sample output of the command::

```
[spacesadmin@connector ~]$ connectorctl -s location macdebug disable -m 34:e1:2d:23:0a:7d

Executing command:macdebug
Command execution status:Success
-----------------------
debug level cleared successfully for mac address: 34:e1:2d:23:0a:7d
```

Command History

Release 3	This command is introduced.

Related Topics

connectorctl -s location macdebug enable, on page 94
connectorctl -s location macdebug viewdebuglogs, on page 92

connectorctl -s location macdebug enable

This command enables debug mode for location service for a particular MAC address. You can then view the debug logs generated for the MAC address using the **connectorctl -s location macdebug viewdebuglogs** command.

connectorctl -s location macdebug enable -m *macaddress* **-l** *level* **-d** *duration*

Syntax Description

Keywords and Variables	Description
-m *macaddress*	MAC address of the Radio Frequency Identification (RFID) tag or the Bluetooth Low Energy (BLE) tag that you want to debug.
-l *level*	Debug level. Values can be any of the following. • MESSAGE: Prints debug messages in human-readable format. • BYTE: Prints debug messages in byte code.
-d *duration*	Debugging time in minutes.

Examples

The following is a sample output of the command::

```
[spacesadmin@connector ~]$ connectorctl -s location macdebug enable -m 34:e1:2d:23:0a:7d
-l MESSAGE -d 5
Executing command:macdebug
Command execution status:Success
-----------------------
debug level set successfully for mac address: 34:e1:2d:23:0a:7d, level: MESSAGE, duration:
 5
```

Command History

Release 3	This command is introduced.

Related Topics

connectorctl -s location macdebug viewdebuglogs, on page 92
connectorctl -s location macdebug disable, on page 93

Weak MAC Commands

connectorctl weakmac reset

To reset the supported list of SSH MAC algorithms on this device, use the **connectorctl weakmac reset** command.

connectorctl weakmac reset

Syntax Description

This command has no keywords or arguments.

Command History

Release 3	This command is introduced.

Examples

The following example shows how to reset the supported list of SSH MAC algorithms on this device. Once reset, you can use the **connectorctl weakmac show** command to verify that the MAC algorithms supported on this device has changed to the default list (including weak MAC algorithms).

```
[spacesadmin@connector ~]$  connectorctl weakmac reset
Executing command:weakmac
Command execution status:Success
-----------------------
Successfully reset weak mac configuration

[spacesadmin@connector3xinteropP83 ~]$ connectorctl weakmac show
Executing command:weakmac
Command execution status:Success
-----------------------
List of supported MAC algorithms is:
macs umac-64-etm@openssh.com,
umac-128-etm@openssh.com,
hmac-sha2-256-etm@openssh.com,
hmac-sha2-512-etm@openssh.com,
hmac-sha1-etm@openssh.com,
umac-64@openssh.com,
umac-128@openssh.com,
hmac-sha2-256,
hmac-sha2-512,
hmac-sha1
```

Related Topics

connectorctl weakmac remove, on page 97
connectorctl weakmac show, on page 98

connectorctl weakmac remove

To remove support for MAC algorithms that are considered weak from the connector configuration, use the **connectorctl weakmac remove** command.

connectorctl weakmac remove

Syntax Description

This command has no keywords or arguments.

Command History

Release 3	This command is introduced.

Examples

The following example shows how to remove weak MAC algorithms from the configuration. You can use the **connectorctl weakmac show** command to verify that there are no weak MAC algorithms in the supported list.

```
[spacesadmin@connector ~]$  connectorctl weakmac remove
Executing command:weakmac
Command execution status:Success
-----------------------
Successfully removed weak mac configuration

[spacesadmin@connector3xinteropP83 ~]$ connectorctl weakmac show
Executing command:weakmac
Command execution status:Success
-----------------------
List of supported MAC algorithms is:
macs umac-128-etm@openssh.com,
hmac-sha2-256-etm@openssh.com,
hmac-sha2-512-etm@openssh.com,
umac-128@openssh.com,
hmac-sha2-256,
hmac-sha2-512
```

Related Topics

connectorctl weakmac reset, on page 96
connectorctl weakmac show, on page 98

connectorctl weakmac show

To show the supported list of all SSH MAC algorithms, use the **connectorctl weakmac show** command.

connectorctl weakmac show

Syntax Description

This command has no keywords or arguments.

Command History

Release 3	This command is introduced.

Examples

The following example shows how to view the list of all supported SSH MAC algorithms. You can see that the list includes a number MAC algorithms that are considered weak.

```
[spacesadmin@connector ~]$  connectorctl weakmac show
Executing command:weakmac
Command execution status:Success
-----------------------
List of supported MAC algorithms is:
macs umac-64-etm@openssh.com,
umac-128-etm@openssh.com,
hmac-sha2-256-etm@openssh.com,
hmac-sha2-512-etm@openssh.com,
hmac-sha1-etm@openssh.com,
umac-64@openssh.com,
umac-128@openssh.com,
hmac-sha2-256,
hmac-sha2-512,
hmac-sha1
```

Related Topics

connectorctl weakmac reset, on page 96
connectorctl weakmac remove, on page 97

Miscellaneous Commands

connectorctl dockersubnet

To configure the IP address of the Docker daemon, use the **connectorctl dockersubnet** command.

connectorctl dockersubnet show

connectorctl dockersubnet add -i *ip-address* **-c** *cidr-length*

Syntax Description

Keywords and Variables	Description
show	Shows the configuration details of the Docker daemon
add	Changes the IP address of the Docker daemon
-i *ip-address* **-c** *cidr-length*	Docker daemon IPv4 or IPv6 address and the Classless interdomain routing (CIDR) length

Command History

Release 3	This command is introduced.

Usage Guidelines

Caution Ensure that the Docker subnet IP used does not overlap with the service-network subnet.

Examples

The following example shows how to change the IP address of the Docker daemon.

```
[spacesadmin@connector ~]$ connectorctl dockersubnet add -i 172.20.10.10 -c 16
Executing command:dockersubnet
Command execution status:Success
-----------------------
Successfully changed the docker subnet configuration to: 172.20.10.10/16
Note: This will restart the docker deamon, so all the connector services, service-manager
will be restarted
```

The following example shows how to view the IP address of the Docker daemon.

```
[spacesadmin@connector ~]$ connectorctl dockersubnet show
Executing command:dockersubnet
Command execution status:Success
-----------------------
{
    "ipv6": true,
    "ip6tables": true,
    "experimental": true,
    "fixed-cidr-v6": "2001:db8:abc1::/64",
    "userland-proxy": false,
    "log-opts": {
        "max-size": "10m",
        "max-file": "5"
    },
    "bip": "172.20.10.10/16"
}
```

connectorctl httpproxy-auth-deny-chars

To update the list of reserved characters supported in proxy passwords, use the **connectorctl httpproxy-auth-deny-chars** command. This command supports only the primary and secondary Ethernet interface.

connectorctl httpproxy-auth-deny-chars { **show** | **add** *character* | **remove** *character* | **reset** }

Syntax Description

Keywords and Variables	Description
show	Shows the list of reserved characters supported in proxy passwords
add *special-character*	Adds support for the specified reserved character in proxy passwords
remove *special-character*	Removes support for the specified reserved character in proxy passwords
reset	Resets the list of reserved characters supported in proxy passwords

Command History

Release 3	This command is introduced.

Examples

The following example shows how to view the list of reserved characters supported in proxy passwords.

```
[spacesadmin@connector ~]$ connectorctl httpproxy-auth-deny-chars show
Executing command:httpproxy-auth-deny-chars
Command execution status:Success
-----------------------
['*', "'", '(', ')', ';', ':', '&', '=', '+', ']', '[', ',', '/', '?', '%']
```

The following example shows how to add support for the specified reserved character '!' in proxy passwords.

```
[spacesadmin@connector ~]$ connectorctl  httpproxy-auth-deny-chars add !
Executing command:httpproxy-auth-deny-chars
Command execution status:Success
-----------------------
Updated reserved list with ['*', "'", '(', ')', ';', ':', '&', '=', '+', ']', '[', ',',
'/', '?', '%', '!']
```

The following example shows how to remove support for the specified reserved character '!' in proxy passwords.

```
[spacesadmin@connector ~]$ connectorctl  httpproxy-auth-deny-chars remove !
Executing command:httpproxy-auth-deny-chars
Command execution status:Success
-----------------------
Remove of the reserved character is successful.
```

The following example shows how to reset the list of reserved characters supported in proxy passwords.

```
[spacesadmin@connector ~]$ connectorctl  httpproxy-auth-deny-chars reset
Executing command:httpproxy-auth-deny-chars
Command execution status:Success
-----------------------
Reset is successful.
```

connectorctl keyexg

To manage the usage of weak key exchange algorithms by the SSH daemon (SSHD), use the **connectorctl keyexg** command.

connectorctl keyexg show

connectorctl keyexg remove { **-a** | **-r** *algorithm-name* }

connectorctl keyexg reset

Command History

Release 3	This command is introduced.

Syntax Description

Keywords and Variables	Description
show	Shows the list of weak key exchange algorithms supported by the SSHD.
remove -a	Removes support for all weak key exchange algorithms from SSHD.
remove -r *algorithm-name*	Removes support for only the specified comma-separated algorithms from SSHD.
reset	Resets the list of key exchange algorithms supported by SSHD.

Examples

The following example shows how to view the list of key exchange algorithms supported by the SSHD.

```
[spacesadmin@connector ~]$ connectorctl keyexg show
Executing command:keyexg
Command execution status:Success
-----------------------
List of supported Key exchange algorithms is:
kexalgorithms curve25519-sha256,curve25519-sha256@libssh.org,ecdh-sha2-nistp256,
ecdh-sha2-nistp384,ecdh-sha2-nistp521,diffie-hellman-group16-sha512,
diffie-hellman-group18-sha512,diffie-hellman-group14-sha1,diffie-hellman-group14-sha256
```

The following example shows how to remove support for all weak key exchange algorithms from SSHD.

```
[spacesadmin@connector ~]$ connectorctl keyexg remove -a
Executing command:keyexg
Command execution status:Success
-----------------------
Removing all unsupported weak algorithms
Successfully removed -diffie-hellman-group-exchange-sha1,diffie-hellman-group1-sha1 key
exchange algorithm(s)
```

The following example shows how to remove support for only the specified comma-separated algorithms from SSHD.

```
[spacesadmin@connector ~]$ connectorctl keyexg remove -r curve25519-sha256
Executing command:keyexg
Command execution status:Success
-----------------------
Successfully removed -curve25519-sha256 key exchange algorithm(s)
```

The following example shows how to reset the list of key exchange algorithms supported by SSHD.

```
[spacesadmin@connector ~]$ connectorctl keyexg reset
Executing command:keyexg
Command execution status:Success
-----------------------
Successfully reset key exchange algorithms configuration
```

connectorctl techsupport

This command gathers and displays technical support information. The command creates a TAR file with information about the network, system, running docker containers, and downloaded images.

connectorctl techsupport

Syntax Description

This command has no keywords or arguments.

Examples

```
[spacesadmin@connector ~]$ connectorctl techsupport
Executing command:techsupport
Command execution status:Success
-----------------------
###############################################################################################
DNA Spaces Connector 3.0 Tech Support Started At: Tue Aug  2 23:46:19 2022
###############################################################################################

Interface Configuration
Ethernet Tool Stats
Ethernet Tool Ring Buffer Sizes
Network Interface Stats
Network Connection Stats
Route Configuration
NTP Stats
NTP Status
DNS Configuration
Domain Information Groper
ARP hosts
SAR Network
File System Usage
Partition Tables
Current Processes
Top Processes
Processor Related Stats
I/O Related Stats
Memory Stats
List Open Files Count
Up Time
SAR CPU
SAR CPU ALL
SAR I/O
SAR Paging and Memory Statistics
SAR Memory Utilization
Docker Downloaded Images
Docker Containers
Docker Service Status
Docker Stats
Service Manager Service Status
Service Agent Service Status
Docker journalctl Status
Connector Service Status
tech support saved to
/home/dnasadmin/techsupport/connector_tech_support_2022-08-02T23-46-19.gz
```

Command History

Release 3	This command is introduced.

connectorctl reset

This command resets all the connector configurations including the HTTP proxy and token.

connectorctl reset

Syntax Description

This command has no keywords or arguments.

Command History

Release 3	This command is introduced.

connectorctl version

This command displays the versions of **service-manager** and **service-agent** services on the connector.

connectorctl version

Syntax Description

This command has no keywords or arguments.

Examples

The following example shows how to view the versions of service-manager and service-agent.

```
[spacesadmin@connector ~]$ connectorctl version
Executing command:version
Command execution status:Success
-----------------------
Package:connector3-p82-sep2022
System Version:8.4.0.82
Service Agent Version:8.4.0.97
Service Manager Version:3.0.1.96
```

Command History

Release 3	This command is introduced.

connectorctl help

This command displays all the commands available on the connector command line interface.

connectorctl help

Syntax Description

This command has no keywords or arguments.

Command History

Release 3	This command is introduced.