# NETCONF and ConfD

This chapter describes NETCONF and the StarOS process called ConfD manager.

It contains the following sections:



## Overview

StarOS provides a northbound NETCONF interface that supports a YANG data model for transferring configuration and operational data with the Cisco Network Service Orchestrator (NSO). It also incorporates a ConfD engine to communicate with the NSO management console.

NETCONF (Network Configuration Protocol) is a network management protocol developed and standardized by the IETF (RFC 6241). It provides mechanisms to install, manipulate, and delete the configuration of network devices. Its operations are realized on top of a simple remote procedure call (RPC) layer. The NETCONF protocol uses XML-based data encoding for the configuration data as well as the protocol messages. The protocol messages are exchanged on top of a secure transport protocol.

ConfD is an on-device management framework that provides a set of interfaces to manage a device. The ConfD framework automatically renders all the management interfaces from a data-model. ConfD implements the full NETCONF specification and runs over SSH with content encoded in XML.

ConfD is configured to allow only authenticated/authorized access through external authentication. ConfD Manager provides a standalone CLI module for ConfD to invoke when authenticating/authorizing any new users. ConfD is configured to allow only authorized access through StarOS authentication. Upon authentication, the user is given a privilege level (0-15) which is mapped to StarOS *secure admin*, *admin*, *operator*, and *inspector*, as defined in the YANG model. StarOS logs CLI authentication event/status messages for each ConfD authentication request.

On the southbound side, ConfD communicates with a StarOS process called via a set of APIs provided by the ConfD management agent. The ConfD Configuration Database (CDB) is used by ConfD to store objects. StarOS accesses the database through the ConfD-supplied APIs. Once the ConfD configuration database is populated, StarOS continues to allow CLI access to modify the overall configuration. There are no automatic updates to the CDB as a result. The CDB only receives updates via the NETCONF interface. In order to keep the CDB and the StarOS configuration databases in sync, all changes made via CLI access (external to NETCONF) to YANG model supported configuration objects must be applied to the CDB manually.

YANG is a data modeling language for the NETCONF network configuration protocol. It can be used to model both configuration data as well as state data of network elements. YANG can also be used to define the format of event notifications emitted by network elements and it allows data modelers to define the signature of remote procedure calls that can be invoked on network elements via the NETCONF protocol (RFC 6020). The YANG file is compiled as part of StarOS and incorporates a subset of the existing StarOS supported CLI commands.

In this release, the YANG model supports a limited set of ECS configuration commands via NSO. For additional information, refer to NSO user documentation and Supported StarOS ECS Configuration Commands, on page 9.

# Configuring ConfD

To enable NETCONF protocol in StarOS, you must enable **server confd** and enter the NETCONF Protocol Configuration mode. The NETCONF Protocol Configuration mode supports optional configuration commands.

## SSH Key Requirement

NETCONF-ConfD support requires that a V2-RSA SSH key be configured on the local context.

If an SSH key is not available, StarOS generates an error message.

```
Failure: The ConfD (NETCONF) server requires an RSA key on the local context
```

You can run the **show ssh key** command to verify the existence of an SSH key on the system.

If an SSH key is not available, see the *Configuring SSH Options* section of the *Getting Started* chapter in this guide.

## NETCONF Protocol Configuration Mode

The NETCONF protocol is enabled via the Context Configuration mode **server conf** command. This command is restricted to the local context only.

```
[local]host_name# configure
[local]host_name(config)# context local
[local]host_name(config-ctx)# server confd
[local]host_name(config-confd)# ?
  autosave-config      - Automatically saves current configuration to the URL specified
                         whenever a change is applied through ConfD interfaces
  confd-user           - Configures the default login user with full administrator rights
                         for the ConfD server.
  end                  - Exits configuration mode and returns to Exec Mode
  exit                 - Exits current configuration mode, returning to previous mode
  no                   - Enables/Disables the followed option
```

The **autosave-config** and **confd-user** keywords are optional.

To disable NETCONF protocol, run the Global Configuration mode **no server confd** command.

For additional information, see the *NETCONF Protocol Configuration Mode Commands* chapter of the *Command Line Interface Reference*.

### autosave-config

This NETCONF Protocol Configuration mode command automatically saves the current ConfD configuration to a specified URL whenever a change is applied by NSO through the ConfD interface in the CLI based model. By default this command is disabled.

The command syntax is: **autosave-config** <*url*>, where **[ file: ]{/flash | /usb1 | /hd-raid | /sftp} [** /<*directory*>**]** /<*filename*>.

### confd-user

This NETCONF Protocol Configuration mode command associates a username for all CLI operations via NETCONF. The user will be authenticated with verifiable credentials. This username is used for CLI logging purposes only.

The command syntax is: **confd-user** <*username*>, where <*username*> is an alphanumeric string of 1 to 144 characters.

**Important** The NETCONF or RESTful session must still be established with verifiable credentials.

## Seeding and Synchronizing the CDB

After enabling **server confd** you may need to initially seed the CDB with a local copy of the configuration database (CDB) managed by ConfD on StarOS. The seeding procedure creates a CDB used by ConfD on the StarOS platform that contains all CLI based YANG model supported configuration commands.

**Important**

- If you manually modify a managed object via the StarOS CLI, you must resynchronize the running configuration with the NSO by repeating the procedure described below.

**Step 1** Run Exec mode **save configuration** <*url*> **confd** to save the ConfD supported StarOS configuration data to a file on the /flash device.

**Step 2** Run Exec mode **show configuration error** to validate the saved configuration. Correct any errors before applying the configuration. Otherwise, ConfD will reject the entire configuration.

**Step 3** Run Exec mode **configure confd** <*url*> to apply the ConfD configuration. Once the ConfD configuration is applied, the device is ready to establish NETCONF connections to the NSO management service.

**Step 4** Synchronize the device with your NSO. Refer to NSO user documentation for detailed information on the synchronization process.

## Sample Configuration

The following command sequence establishes a COnfD configuration in support of NETCONF protocol.

A type v2-RSA SSH key is required for enabling **server confd**.

```
configure
  context local
      ssh key
<encrypted key text>
len 938 type v2-rsa
    server confd
      confd-user NETCONF
      autosave-config /flash/config.cfg
    #exit
    subscriber default
    exit
    aaa group default
    #exit
    gtpp group default
    #exit
  #exit
end
```

Notes:

- **confd-user** and **autosave-config** are optional. Just configuring **server confd** enables NETCONF support.

# Verifying the Configuration

There are two Exec mode **show** commands that display information about the NETCONF-ConfD configuration.

## show confdmgr Command

This command displays information about the StarOS ConfD Manager (confdmgr) process.

The syntax for this command is:

**show confdmgr { confd { cdb | netconf | state } | subscriptions }**

Notes:

- The **confd** keyword displays the following options:
  - **cdb** displays ConfD CDB information
  - **netconf** displays NETCONF state information
  - **state** displays current ConfD state information
- The **subscriptions** keyword displays ConfD CDB subscription information.

See below for a sample output for **show confdmgr**:

```
[local]<host_name># show confdmgr
```

```
State Information
-----------------
State               Started
Subscriptions       2
Last successful id  1461-704882-705350
Last failed id      None
Autosave url        Not configured
Username            Not configured

Statistics
----------
Triggers                  1
Notifications             2
Successful notifications  2
Failed notifications      0
Unexpected                0
[local]<host_name>#
```

The Statistics portion of this output includes the following information:

- **Triggers –** Number of times confdmgr has requested ConfD to dump the CDB contents back into confdmgr which results in a config synchronization by SCT (Shared configuration Task).
- **Notifications –** Number of times ConfD has sent an update to confdmgr.
- **Successful Notifications –** Number of times an update received from ConfD was successfully processed.
- **Failed Notifications –** Number of times an update received from ConfD was not processed successfully. The number of successes and failures should always equal the total number of notifications.
- **Unexpected –** Number of times an unexpected condition was encountered. An error log is generated for each case.

See below for a sample output for **show confdmgr confd cbd**:

```
[local]<host_name># show confdmgr confd cdb
active-charging service acs
exit
context local
 server confd
  autosave-config /tmp/ut_confdmgr_config.txt
  confd-user      confd_user
 exit
exit
nacm read-default permit
nacm groups group admin
!
nacm groups group inspector
!
nacm groups group operator
!
nacm groups group secure_admin
!
nacm rule-list secure_admin
!
 group [ secure_admin ]
 rule any-access
  action permit
!
 rule secure_admin_server_confd
  module-name       cisco-staros-cli-config
  path              /context/server/confd
  access-operations create,read,update
  action            permit
  |
  |
  V
nacm rule-list inspector
 group [ inspector ]
 rule any-access
```

```
  access-operations read
  action            permit
 !
!
[local]<host_name>#
```

See below for a sample output for **show confdmgr confd netconf**:

```
[local]<host_name># show confdmgr confd netconf
netconf-state capabilities capability urn:ietf:params:netconf:base:1.0
netconf-state capabilities capability urn:ietf:params:netconf:base:1.1
netconf-state capabilities capability urn:ietf:params:netconf:capability:writable-running:1.0
netconf-state capabilities capability urn:ietf:params:netconf:capability:candidate:1.0
 |
 |
 V
netconf-state statistics netconf-start-time 2016-03-30T17:09:49-04:00
netconf-state statistics in-bad-hellos 0
netconf-state statistics in-sessions 0
netconf-state statistics dropped-sessions 0
netconf-state statistics in-rpcs 0
 |
 |
 V
netconf-state datastores datastore candidate
NAME         CREATOR  CREATED                    CONTEXT
-----------------------------------------------------------
/rollback0   admin    2016-04-26T17:08:02-00:00  noaaa
/rollback1   admin    2016-04-26T17:07:57-00:00  noaaa
/rollback2   admin    2016-04-26T17:07:49-00:00  noaaa
/rollback3   admin    2016-04-26T17:07:49-00:00  noaaa
/rollback4   admin    2016-04-26T17:07:47-00:00  noaaa
 |
 |
 V
/cli-history/admin.hist
/cli-history/root.hist
/global.data

[local]<host_name>#
```

See below for a sample output for **show confdmgr confd state**:

```
[local]<host_name># show confdmgr confd state
confd-state version 6.1.1
confd-state epoll false
confd-state daemon-status started
confd-state loaded-data-models data-model cisco-staros-cli-confi
 revision        2016-03-15
 namespace       http://www.cisco.com/staros-cli-config
 prefix          staros_cli
 exported-to-all
confd-state loaded-data-models data-model iana-crypt-hash
 revision        2014-04-04
 namespace       urn:ietf:params:xml:ns:yang:iana-crypt-hash
 prefix          ianach
 exported-to-all
confd-state loaded-data-models data-model ietf-inet-types
 revision        2013-07-15
 namespace       urn:ietf:params:xml:ns:yang:ietf-inet-types
 prefix          inez
 exported-to-all
confd-state loaded-data-models data-model ietf-netconf-acm
 revision        2012-02-22
 namespace       urn:ietf:params:xml:ns:yang:ietf-netconf-acm
 prefix          nacm
 exported-to-all
 |
 |
 V
NETCONF TCP listen addresses:
IP         PORT
-----------------
127.0.0.1  2023
```

```
NETCONF SSH listen addresses:
IP       PORT
0.0.0.0  2022

CLI SSH listen addresses:
IP       PORT
---------------
0.0.0.0  2024

confd-state internal cdb datastore running
 transaction-id               1461-704882-705350
 filename                     /root/sandbox/confd/install/var/confd/cdb/A.cdb
 ram-size                     "4.44 KiB"
 read-locks                   0
 write-lock-set               false
 waiting-for-replication-sync false
confd-state internal cdb client
 name confdmgr
 info 12951/4
 type subscriber
 subscription
  datastore running
  priority  0
  id        7
  path      /context
 subscription
  datastore running
  priority  0
  id        6
  path      /active-charging
local]<host_name>#
```

See below for a sample output for **show confdmgr subscriptions**:

```
[local]<host_name># show confdmgr subscriptions

Subscriptions:
Path                                Index  Namespace
-------------------------------------------------
/active-charging                        6     staros
/context                                7     staros
[local<host_name>#
```

Subscriptions are configuration points defined in the Yang model for which confdmgr wants to be notified when a change occurs. In this release there are two subscriptions: “/active-charging” and “/context”.

# show configuration confd Command

The **confd** keyword filters the output of the **show configuration** command to display only configuration commands that are supported by the YANG model.

```
show configuration confd
```

A sample output appears below.

```
[local]<host_name># show configuration confd
config
  context local
    server confd
    #exit
  active-charging service ecs
    ruledef rd1
      tcp any-match = TRUE
    #exit
    rulebase default
    #exit
  #exit
end
[local]<host_name>#
```

## clear confdmgr statistics

This command clears everything listed in the "Statistics" section of the output of the **show confdmgr** command, including:

- Triggers
- Notifications
- Successful notifications
- Failed notifications
- Unexpected

# Show Support Details (SSD)

The output of all **show confdmgr** commands has been added to the SSD.

# CDB Maintenance

A local copy of the ConfD Configuration Database (CDB) is managed by ConfD on StarOS.

You can show and save all ConfD supported StarOS configuration commands to a URL. The **confd** keyword has been added to the **show configuration** and **save configuration** commands for these purposes.

After saving a ConfD-supported configuration to a URL, you can apply it directly to the CDB via the Exec mode **configure confd** <*url*> command. This command applies the contents of the file at the *url* to the running configuration of ConfD.

Additional detail regarding the above commands is provided below.

## clear confdmgr confd cdb

This Exec mode command erases the configuration in the ConfD Configuration Database (CDB) which is used by ConfD to store configuration objects. StarOS accesses the database via ConfD-supplied APIs.

**Note** The CDB cannot be erased unless the Context Configuration mode **no server confd** command is run in the local context to disable ConfD and NETCONF protocol support.

The following is a sample command sequence for clearing the CDB:

```
[local]host_name# config
[local]host_name(config)# context local
[local]host_name(config-ctx)# no server confd
[localhost_name(config-ctx)# end
[local]host_name# clear confdmgr confd cdb
About to delete the ConfD configuration database
The running configuration is NOT affected.
```

```
Are you sure? [Yes|No]: y
[local]host_name#
```

**Caution** Clearing the CDB is a terminal operation. The CDB must be repopulated afterwards.

## configure confd <url>

This Exec mode command applies the contents of the configuration script specified by the URL to the current ConfD configuration database (CDB).

A sample command sequence is provided below.

```
[local]host_name# save configuration /flash/confd.config confd
[local]host_name# configure confd /flash/confd.config
Info: #!$$ StarOS V20.2 Chassis 52767e9ff9e207bed12c76f7f8a5352c
Info: config
Info:   active-charging service acs
Info:     rulebase default
Info:     #exit
Info:   #exit
Info: end
[local]host_name#
```

## save configuration <url> confd

The keyword **confd** is added to the Exec mode **save configuration** command. This keyword filters the saved configuration commands to contain only configuration commands that are supported by the YANG model.

The command syntax for this process is:

`[local]host_name#` **save configuration** *<url>* **confd**

The output of the YANG model subset of configuration commands can be viewed via the **show file url** *<url>* command, where *<url>* is the pathname used to save the configuration. The saved configuration file can then be applied to the CDB using the **configure confd** command.

# Supported StarOS ECS Configuration Commands

For this release, the following StarOS ECS commands are supported for the CLI based YANG model:

- ruledef *<ruledef_name>*
  - ip server-ip-address = *
  - tcp-ether-port = *
  - udp ether-port = *
  - tcp either-port-range = *
  - udp ether-port range = *
  - tcp-any-match = *
  - udp any-match = *

    - http url = *
    - httpcookie = *
    - http x-header = *
- group-of-ruledefs <*ruledefs_group_name*>
    - add-ruledef priority = *
- qos-group-of-ruledefs <*group_name*>
    - add-group-of-ruledef <*group_of_ruledef_name*>
- charging-action <*charging_action_name*>
    - flow-idle-timeout <*seconds*>
    - content-id 1
    - service-identifier <*service_id*>
    - billing-action egcdr
- rulebase <*rulebase_name*>
    - action priority <*priority_number*> group-of-ruledefs <*ruledefs_group_name*> charging-action <*charging_action_name*>

**Note** "= *" indicates support for every option following the prior keyword/value.