NETCONF and ConfD

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

It contains the following sections:

Feature Summary and Revision History

Summary Data

Applicable Product(s) or Functional Area	All
Applicable Platform(s)	ASR 5500 VPC-SI VPC-DI
Feature Default	Disabled - Configuration Required
Related Changes in This Release	Not Applicable

Related Documentation	• *ASR 5500 System Administration Guide* • *Command Line Interface Reference* • *VPC-DI System Administration Guide* • *VPC-SI System Administration Guide*

Revision History

Note Revision history details are not provided for features introduced before releases 21.2 and N5.5.

Revision Details	Release
SNMP MIB alerts and alarms are now able to be sent via NETCONF notifications. The **netconf** command in NETCONF Protocol Configuration Mode added a **snmp** keyword to enable this functionality. **show confdmgr** command output expanded.	21.3
ConfD may now collect bulkstats operational data that is retrieved via REST interface. New StarOS bulkstats and server ConfD configuration YANG models are supported. Any updates via StarOS CLI are now automatically synced back to the ConfD Database. The CLI based YANG model is only applicable to StarOS ECS (Enhanced Charging System) commands. NETCONF Protocol Configuration Mode added **bulkstats**, **netconf**, and **rest** commands. **autosave-config** command obsoleted. **show confdmgr**command added keywords **model bulkstats** and **model confd**. **show confdmgr** command output expanded.	21.2
First introduced.	Pre 21.2

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 manager (confdmgr) 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. The confdmgr 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.

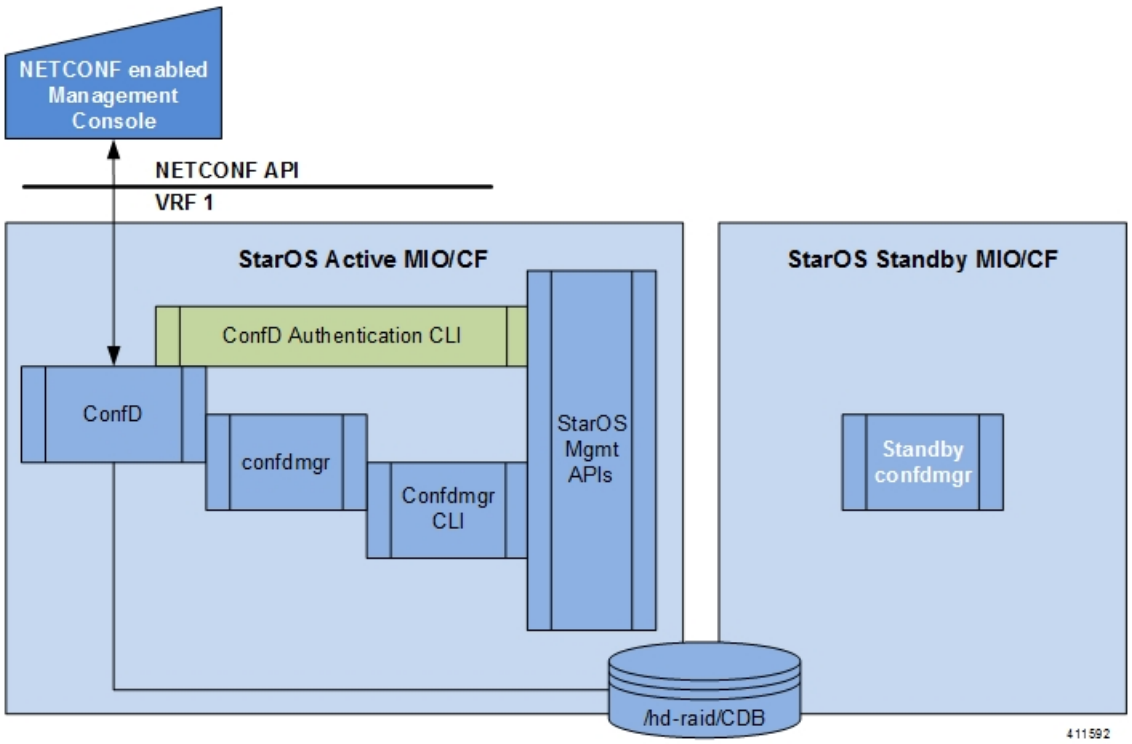
On the southbound side, ConfD communicates with a StarOS process called via a set of APIs provided by the ConfD management agent. The ConfD Configuration Database (CDB) is used by ConfD to store objects. StarOS accesses the database through the ConfD-supplied APIs. Any updates via StarOS CLI are automatically synced back to the CDB.

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 existing StarOS supported CLI commands.

ConfD may also collect bulkstats operational data. When enabled, StarOS will send schema information to confdmgr while gathering statistics. Collected bulkstats are stored in the ConfD CDB for later retrieval over REST (Representational State Transfer) interface. RESTCONF is an IETF draft (draft-bierman-netconf-restconf-4) that describes how to map a YANG specification to a RESTful interface using HTTP as transport. REST and RESTCONF are only enabled internally when a valid certificate and key are configured. If client authentication is enabled, CA-certificates may be required as well.

For additional NSO information, refer to the NSO user documentation.

Figure 1: NETCONF System Flow



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)# ?
  bulkstats            - Populate ConfD with bulkstats operational data
  confd-user           - Configures the default login user with full administrator rights
                         for the ConfD server.
  do                   - Spawns an exec mode command which displays
                         information to the administrator
  end                  - Exits configuration mode and returns to Exec Mode
  exit                 - Exits current configuration mode, returning to previous mode
  netconf              - Configure the netconf interface
  no                   - Enables/Disables the followed option
  rest                 - Configure the rest interface
```

The following keywords are optional:

- **bulkstats**
- **confd-user**
- **netconf**
- **rest**

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

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

bulkstats

This NETCONF Protocol Configuration mode command enables bulkstats collection and reporting via REST interface. By default, this command is disabled.

The command syntax is: **bulkstats**.

During StarOS statistics gathering, bulk statistics are also stored in the CDB for later retrieval over REST interface.

Use **no bulkstats** to disable populating ConfD with bulkstats operational data.

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

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.

netconf notifications events

This NETCONF Protocol Configuration mode command enables events logged in StarOS to be sent out as NETCONF notifications on the stream named "StarOS." Level specifies the lowest event severity level that results in a notification.

The command syntax is: **netconf notifications events level { critical | error | warning | unusual | info }**, where

- **critical** - Level 1: Reports critical errors contained in log file.
- **error** - Level 2: Reports error notifications contained in log file.
- **warning** - Level 3: Reports warning messages contained in log file.
- **unusual** - Level 4: Reports unexpected errors contained in log file.
- **info** - Level 5: Reports informational messages contained in log file.

Use **no netconf notifications events** to disable NETCONF notifications.

Important Any event that is of category "critical-info" (regardless of severity) will also be converted to notifications.

netconf notifications snmp

This NETCONF Protocol Configuration mode command enables SNMP alerts and alarms to be sent out as NETCONF notifications on the stream named "StarOS_SNMP".

The command syntax is: **netconf notifications snmp**.

Use **no netconf notifications snmp** to disable NETCONF notifications.

netconf port

This NETCONF Protocol Configuration mode command sets the NETCONF interface port number. When **server confd** is enabled, the default port is automatically set to 830.

The command syntax is: **netconf port** *port_number*, where *port_number* must be an integer from 1 through 65535.

Use **no netconf port** to reset the port number to 830.

Important A change to the NETCONF interface port value will result in a planned restart of ConfD and temporary loss of connectivity over the NETCONF and REST (if enabled) interfaces.

rest auth-policy

This NETCONF Protocol Configuration mode command controls the level of verification the server does on client certificates. CA (certificate authority) certificates can be configured using the existing **ca-certificate** command in Global Configuration mode.

The command syntax is: **rest auth-policy { none | peer | peer-fail }**, where

- **none** - No authentication performed.
- **peer** - If the client does not provide a certificate, or the client provides a certificate and it is valid, the connection is allowed. If the client provides a certificate that is not valid, the connection is aborted.

Important If **peer** is selected, CA certificates are recommended; otherwise, a client providing a valid certificate cannot be authenticated and connection will fail.

- **peer-fail** - Server requires the client to supply a client certificate and will fail the connection if certificate is not successfully validated.

Important If **peer-fail** is selected, one or more CA certificates must be present on the device; otherwise, the REST interface will not be enabled.

Use **no rest auth-policy** to set the auth-policy to **none**; no authentication will be performed.

Important A change to the REST interface auth-policy may result in a planned restart of ConfD and temporary loss of connectivity over the NETCONF and REST (if still enabled) interfaces.

Changes to global certificates which ConfD is using while REST is enabled will also result in a restart of ConfD.

rest certificate

This NETCONF Protocol Configuration mode command configures certificate and private-key for REST interface.

The command syntax is: **rest certificate** *certificate_name*, where *certificate_name* is an alphanumeric string of 1 to 128 characters.

Important The certificate specified must to be present on the device. Certificate and the associated private-key can be configured using the existing **certificate** command in Global Configuration mode.

Use **no rest certificate** to remove any configured certificate and key. REST will not be operational without a valid certificate and key.

Important A change to the REST interface certificate may result in a planned restart of ConfD and temporary loss of connectivity over the NETCONF and REST (if still enabled) interfaces.

Changes to global certificates which ConfD is using while REST is enabled will also result in a restart of ConfD.

rest hostname

This NETCONF Protocol Configuration mode command specifies a hostname the web server will serve. If configured, mandates the web server to only service requests whose Host field matches the configured hostname.

The command syntax is: **rest hostname** *host_name*, where *host_name* is an alphanumeric string of 1 to 63 characters.

Use **no rest hostname** to use the system name; matching of hostname is not mandated.

Important A change to the REST interface hostname may result in a planned restart of ConfD and temporary loss of connectivity over the NETCONF and REST (if still enabled) interfaces.

Changes to global certificates which ConfD is using while REST is enabled will also result in a restart of ConfD.

rest port

This NETCONF Protocol Configuration mode command sets the REST interface port number.

The command syntax is: **rest port** *port_number*, where *port_number* must be an integer from 1 through 65535.

Use **no rest port** to reset the port number to default 443.

Important A change to the REST interface port value may result in a planned restart of ConfD and temporary loss of connectivity over the NETCONF and REST (if still enabled) interfaces.

Changes to global certificates which ConfD is using while REST is enabled will also result in a restart of ConfD.

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
```

```
      bulkstats
      confd-user NETCONF
      rest certificate rest-cert
    #exit
    subscriber default
    exit
    aaa group default
    #exit
    gtpp group default
    #exit
  #exit
end
```

Notes:

- **bulkstats**, **confd-user**, and **rest** 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 } | model { bulkstats | confd } | subscriptions] [| { grep *grep_options* **| more }]**

Notes:

- The **confd** keyword displays information about the ConfD engine based on the specified keyword in the following options:
  - **cdb** displays ConfD CDB information
  - **netconf** displays NETCONF state information
  - **state** displays current ConfD state information
- The **model** keyword displays information about the ConfD model based on the specified keyword in the following options:
  - **bulkstats** bulk statistics configuration and operational data
  - **confd** server ConfD configuration
- The **subscriptions** keyword displays ConfD CDB subscription information.

show confdmgr

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

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

State Information
-----------------
State               Started
```

```
Subscriptions       5
Last successful id  1461-704882-705350
Last failed id      None
Username            Not configured
Bulkstats                          Enabled
Event notification level           Disabled
SNMP notifications                 Disabled
REST interface authentication      none
REST interface certificate         rest-cert
REST interface host name           Not configured

Interface                Status           Port
-----------------------------------------------
NETCONF                  Enabled           830
REST                     Enabled           443

Statistics
----------
Triggers                      1
Replays                       0
Notifications                 5
Notification failures         0
Trigger failures              0
Replay failures               0
NETCONF notification failures 0
Unexpected failures           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 configuration synchronization to SCT (Shared Configuration Task).
- **Replays –** Number of times a transaction has been replayed. A replay is initiated if, upon startup, the last successful transaction ID in confdmgr does not match that of ConfD. This could occur, for example, if confdmgr task restarted when processing the notification for a configuration transaction.
- **Notifications –** Number of times ConfD has sent a configuration update to confdmgr. For example, this can occur as the result of a "commit" via confd_cli or during a trigger event.
- **Notification failures –** Number of times configuration received from ConfD was not processed successfully.
- **Trigger failures –** Number of times a CDB dump to confdmgr failed.
- **Replay failures –** Number of times an attempt to replay a transaction failed.
- **NETCONF notification failures –** Number of times an attempt to issue a NETCONF notification failed.
- **Unexpected failures –** Number of times an unexpected condition was encountered. An error log is generated for each case.

show confdmgr confd cdb

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

```
[local]<host_name># show confdmgr confd cdb
bulkstats server collection true
bulkstats server historical-collection false
bulkstats server gather-on-standby true
bulkstats server sample-interval 60
bulkstats server transfer-interval 1440
bulkstats server limit 7500
bulkstats server receiver-mode secondary-on-failure
bulkstats server file 1
!
bulkstats schemas file 1
```

```
 schema-type system
  schema abc
   format      %host%
   active-only false
  !
  schema common
   format      %host%,%ipaddr%,%time%,%uptime%,%swbuild%,%localtz%
   active-only false
  !
  schema systemSch11
   format
PPM,system,systemSch11,%epochtime%,%localdate%,%localtime%,%uptime%,%diamauth-msg
-saans%,%diamauth-msg-sarretry%,%diamauth-msg-saatimeout%,%diamauth-msg-saadropped%,%diamauth-ms
g-uareq%,%diamauth-msg-uaans%,%diamauth-msg-uarretry%,%diamauth-msg-uaatimeout%,%diamauth-msg-ua
adropped%,%diamauth-msg-lireq%,%diamauth-msg-lians%,%diamauth-msg-lirretry%,%diamauth-msg-liatim
eout%,%diamauth-msg-liadropped%,%diamauth-msg-rtreq%,%diamauth-msg-rtans%,%diamauth-msg-rtrrejec
t%,%diamauth-msg-ppreq%,%diamauth-msg-ppans%,%diamauth-msg-pprreject%,%diamauth-msg-dereq%
   active-only false
  !
 !
!
confd bulkstats true
confd netconf port 830
confd rest port 443
confd rest auth-policy none
confd rest certificate rest-cert
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>#
```

show confdmgr confd netconf

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
netconf-state schemas schema cisco-staros-bulkstats 2016-12-14 yang
 namespace http://www.cisco.com/staros-bulkstats
 location  [ NETCONF ]
netconf-state schemas schema cisco-staros-bulkstats-config 2016-12-14 yang
 namespace http://www.cisco.com/staros-config
 location  [ NETCONF ]
 |
 |
 V
NAME                      CREATOR  CREATED                     CONTEXT
-------------------------------------------------------------------------
/rollback0                system   2017-01-17T13:40:53-00:00  system
/rollback1                system   2017-01-17T13:40:52-00:00  system
/rollback2                system   2017-01-17T13:40:52-00:00  system
/rollback3                system   2017-01-17T13:40:52-00:00  system
/rollback4                system   2017-01-17T13:36:43-00:00  system
 |
 |
 V
/cli-history/admin.hist
/cli-history/root.hist
/global.data

netconf-state streams stream NETCONF
 description              "default NETCONF event stream"
 replay-support           false
netconf-state streams stream StarOS
 description              "StarOS Notifications"
 replay-support           true
 replay-log-creation-time 2017-02-10T16:00:59+00:00
[local]<host_name>#
```

show confdmgr confd state

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

```
[local]<host_name># show confdmgr confd state
confd-state version 6.3
confd-state epoll false
confd-state daemon-status started
confd-state loaded-data-models data-model cisco-staros-bulkstats
 revision        2016-12-14
 namespace       http://www.cisco.com/staros-bulkstats
 prefix          staros_bulkstats
 exported-to-all
confd-state loaded-data-models data-model cisco-staros-cli-config
 revision        2016-12-14
 namespace       http://www.cisco.com/staros-cli-config
 prefix          staros_cli
 exported-to-all
confd-state loaded-data-models data-model cisco-staros-config
 revision        2016-12-14
 namespace       http://www.cisco.com/staros-config
 prefix          staros_config
 exported-to-all
confd-state loaded-data-models data-model cisco-staros-exec
 revision        2016-12-14
 namespace       http://www.cisco.com/staros-exec
 prefix          staros_exec
 exported-to-all
confd-state loaded-data-models data-model cisco-staros-notif
 revision        2016-12-14
 namespace       http://www.cisco.com/staros-notif
 prefix          staros_notif
```

```
 exported-to-all
confd-state loaded-data-models data-model iana-crypt-hash
 revision        2014-08-06
 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          inet
 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
confd-state loaded-data-models data-model ietf-netconf-monitoring
 revision        2010-10-04
 namespace       urn:ietf:params:xml:ns:yang:ietf-netconf-monitoring
 prefix          ncm
 exported-to-all
confd-state loaded-data-models data-model ietf-netconf-notifications
 revision        2012-02-06
 namespace       urn:ietf:params:xml:ns:yang:ietf-netconf-notifications
 prefix          ncn
 exported-to-all
confd-state loaded-data-models data-model ietf-restconf-monitoring
 revision        2016-08-15
 namespace       urn:ietf:params:xml:ns:yang:ietf-restconf-monitoring
 prefix          rcmon
 exported-to-all
confd-state loaded-data-models data-model ietf-yang-library
 revision    2016-06-21
 namespace   urn:ietf:params:xml:ns:yang:ietf-yang-library
 prefix          yanglib
  exported-to [ netconf rest ]
confd-state loaded-data-models data-model ietf-yang-types
 revision        2013-07-15
 namespace       urn:ietf:params:xml:ns:yang:ietf-yang-types
 prefix          yang
 exported-to-all
confd-state loaded-data-models data-model netconf_netmod
 namespace   urn:ietf:params:xml:ns:netmod:notification
 prefix      nm
 exported-to [ netconf ]
confd-state loaded-data-models data-model tailf-aaa
 revision        2015-06-16
 namespace       http://tail-f.com/ns/aaa/1.1
 prefix          aaa
 exported-to-all
confd-state loaded-data-models data-model tailf-acm
 revision        2013-03-07
 namespace       http://tail-f.com/yang/acm
 prefix          tacm
 exported-to-all
confd-state loaded-data-models data-model tailf-common-monitoring
 revision        2013-06-14
 namespace       http://tail-f.com/yang/common-monitoring
 prefix          tfcg
 exported-to-all
confd-state loaded-data-models data-model tailf-confd-monitoring
 revision        2013-06-14
 namespace       http://tail-f.com/yang/confd-monitoring
 prefix          tfcm
 exported-to-all
confd-state loaded-data-models data-model tailf-kicker
 revision        2016-11-24
 namespace       http://tail-f.com/ns/kicker
 prefix          kicker
 exported-to-all
confd-state loaded-data-models data-model tailf-netconf-monitoring
 revision        2016-11-24
```

```
 namespace       http://tail-f.com/yang/netconf-monitoring
 prefix          tncm
 exported-to-all
confd-state loaded-data-models data-model tailf-rollback
 revision    2016-09-15
 namespace   http://tail-f.com/ns/rollback
 prefix          rollback
 exported-to [ rest ]
confd-state loaded-data-models data-model tailf-webui
 revision        2013-03-07
 namespace       http://tail-f.com/ns/webui
 prefix          webui
 exported-to-all
NETCONF SSH listen addresses:
IP       PORT
---------------
0.0.0.0  830

CLI SSH listen addresses:
IP         PORT
-----------------
127.0.0.1  2024

WebUI SSL listen addresses:
IP       PORT
---------------
0.0.0.0  443

REST SSL listen addresses:
IP       PORT
---------------
0.0.0.0  443

confd-state internal callpoints actionpoint StarOSexec
 daemon id 0
 daemon name confdmgr
confd-state internal callpoints notification-stream-replay NETCONF
 replay-support none
confd-state internal callpoints notification-stream-replay StarOS
 replay-support builtin
confd-state internal cdb datastore running
 transaction-id              1484-678453-229261
 filename                    /hd-raid/confd_dir/var/confd/cdb/A.cdb
 disk-size                   "3.16 KiB"
 ram-size                    "9.43 KiB"
 read-locks                  0
 write-lock-set              false
 waiting-for-replication-sync false
confd-state internal cdb datastore operational
 filename              /hd-raid/confd_dir/var/confd/cdb/O.cdb
 disk-size             "4 bytes"
 ram-size              "6.99 KiB"
 subscription-lock-set false
confd-state internal cdb client
 name confdmgr
 info 5420/10
 type subscriber
 subscription
  datastore running
  priority  -2147483648
  id        7
  path      /context
 subscription
  datastore running
  priority  -2147483648
  id        6
  path      /active-charging
local]<host_name>#
```

show confdmgr model bulkstats

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

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

Model: Bulkstats
----------------

Operational Data:
  Requests               277
  Records                831
  Failures               0

Configuration:
  CLI updates            0
  NETCONF updates        2
  Aborts                 0
  Failures               0
local]<host_name>#
```

The Operational Data portion of this output includes the following information:

- **Requests –** Number of operational data msg requests from bulkstats to confdmgr.
- **Records –** Number of operational data schema records processed.
- **Failures –** Number of errors detected in confdmgr while processing push requests from bulkstats.

The Configuration portion of this output includes the following information:

- **CLI updates –** Number of push configuration requests from the CLI as well as configuration loads from SCT.
- **NETCONF updates –** Number of bulkstats subscription notifications.
- **Aborts –** Number of times a configuration update via NETCONF was aborted.
- **Failures –** Number of errors detected processing any bulkstats configuration requests within confdmgr.

show confdmgr model confd

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

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

Model: ConfD
----------------
CLI updates       0
NETCONF updates   1
Aborts            0
Failures          0
local]<host_name>#
```

- **CLI updates –** Number of push configuration requests from the CLI as well as configuration loads from SCT.
- **NETCONF updates –** Number of ConfD configuration subscription notifications.
- **Failures –** Number of errors detected processing any ConfD configuration requests within confdmgr.
- **Aborts –** Number of times a configuration update via NETCONF was aborted.

show confdmgr subscriptions

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

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

Subscriptions:
Path                                 Index  Namespace
-------------------------------------------------
/active-charging                         6     http://www.cisco.com/staros-cli-con
fig
/context                                 7     http://www.cisco.com/staros-cli-con
fig
/bulkstats/server                        8     http://www.cisco.com/staros-config
/bulkstats/schemas                       9     http://www.cisco.com/staros-config
/confd                                  10     http://www.cisco.com/staros-config
[local<host_name>#
```

Subscriptions are configuration points defined in the Yang model for which confdmgr wants to be notified when a change occurs.

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 will be repopulated when the Context Configuration mode **server confd** command is run in the local context to re-enable ConfD and NETCONF protocol support.

clear confdmgr statistics

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

- Triggers
- Replays

- Notifications
- Notification failures
- Trigger failures
- Replay failures
- NETCONF notification failures
- Unexpected failures

YANG Models

The following YANG files are available in the StarOS installation:

- **cisco-staros-bulkstats-config.yang** - StarOS native bulkstats configuration model.
- **cisco-staros-bulkstats-schema-types.yang** - An extension to the **cisco-staros-bulkstats-config.yang** model that contains an enumerated list of schema names pulled directly from the code.
- **cisco-staros-bulkstats.yang** - Operational data model that enables customers to obtain bulk statistics via the RESTful interface. Only users with admin credentials may use this model.
- **cisco-staros-cli-config.yang** - Obsolete CLI-centric model introduced in StarOS release 20.2; model no longer supported.
- **cisco-staros-confd-config.yang** - Native server ConfD configuration model.
- **cisco-staros-config.yang** - Container yang file used to include all other cisco-staros-* configuration models (all native models are included here under a common namespace).
- **cisco-staros-exec.yang** - Model to enable CLI exec operations via the restful interface. Only users with admin credentials may use this model. Used by ConfD locally to parse input.
- **cisco-staros-notif.yang** - Model to enable NETCONF notification streams for StarOS event logging. Debug level events are not supported; only informational messages and above are supported.

Important The ConfD server must be started at least once before these YANG files are populated and available.

YANG files must be pulled to the Cisco NSO to build StarOS Network Element Drivers (NEDs).

To copy YANG files, enter commands similar to the following:

```
#copy /hd-raid/confd_dir/etc/confd/cisco-staros-confd-config.yang
sftp://<user>:<password>@<host>/sftp-directory/cisco-staros-confd-config.yang
```

Show Support Details (SSD)

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

ConfD Examples

Server ConfD

The following examples use full TLS authentication and curl to obtain server ConfD configuration.

Server ConfD Configuration

See below for a sample configuration for server ConfD with RESTful interface enabled using non-default NETCONF and HTTPS ports:

```
[local]<host_name># show configuration confd
[local]<host_name># config
[local]<host_name>(config)# ca-certificate name ca-cert pem url /flash/ssl/rootCA.pem
[local]<host_name>(config)# certificate name rest-cert pem url /flash/ssl/host.crt private-key
 pem url /flash/ssl/host.key
[local]<host_name>(config)# end
[local]<host_name># config
[local]<host_name>(config)# context local
[local]<host_name>(config-ctx)# server confd
[local]<host_name>(config-confd)# netconf port 123
[local]<host_name>(config-confd)# rest port 234
[local]<host_name>(config-confd)# rest certificate rest-cert
[local]<host_name>(config-confd)# rest auth-policy peer-fail
[local]<host_name>(config-confd)# end
[local]<host_name># show confdmgr

State Information
 -----------------
State                           Started
Subscriptions                   5
Last successful id              1488-211047-99241
Last failed id                  None
Username                        Not configured
Bulkstats                       Disabled
Event notification level        Disabled
SNMP notifications              Disabled
REST interface authentication   peer-fail
REST interface certificate      rest-cert
REST interface host name        Not configured

Interface               Status            Port
 ---------------------------------------------
NETCONF                 Enabled            123
REST                    Enabled            234

Statistics
----------
Triggers                     1
Replays                      0
Notifications                8
Notification failures        0
Trigger failures             0
Replay failures              0
NETCONF notification failures 0
Unexpected failures          0
```

Using Netconf-console to Obtain the Server ConfD Configuration

See below for a sample use of netconf-console to obtain the server ConfD configuration via NETCONF:

```
 [user@server]$ ./netconf-console --host 1.2.3.4 -u admin --password pswd! --port 123
--get-config -x confd
 <?xml version="1.0" encoding="UTF-8"?>
```

```
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1">
   <data>
       <confd xmlns="http://www.cisco.com/staros-config">
           <bulkstats>false</bulkstats>
           <netconf>
               <port>123</port>
           </netconf>
           <rest>
               <port>234</port>
               <auth-policy>peer-fail</auth-policy>
               <certificate>rest-cert</certificate>
           </rest>
       </confd>
   </data>
</rpc-reply>
```

Notes:

- netconf-console is freely available from GitHub (https://github.com/tail-f-systems/JNC/blob/master/examples/2-junos/netconf-console).

Using Curl to Obtain the Server ConfD Configuration

See below for a sample use of curl to perform the same **get-config** operation:

```
 [<user>@server] ]$ curl -u admin:pswd!
https://rtp-mitg-si06.cisco.com:234/api/running/confd?deep --cert
/users/<user>/ssl_cert/client_cert/client.crt --key
/users/<user>/ssl_cert/client_cert/client.key --cacert
/users/<user>/ssl_cert/root_cert/rootCA.pem

<confd xmlns="http://www.cisco.com/staros-config" xmlns:y="http://tail-f.com/ns/rest"
xmlns:staros_config="http://www.cisco.com/staros-config">
   <bulkstats>false</bulkstats>
   <netconf>
      <port>123</port>
   </netconf>
   <rest>
      <port>234</port>
      <auth-policy>peer-fail</auth-policy>
      <certificate>rest-cert</certificate>
   </rest>
</confd>
```

Bulkstats

The following examples show bulk statistics operational data.

Enable Bulkstats

Enable bulkstats under server ConfD:

```
[local]<host_name># config
[local]<host_name>(config)# context local
[local]<host_name>(config-ctx)# server confd
[local]<host_name>(config-confd)# bulkstats
[local]<host_name>(config-confd)# end
[local]<host_name># show confdmgr

State Information
-----------------
State                           Started
Subscriptions                   5
Last successful id              1488-216669-170664
Last failed id                  None
Username                        Not configured
Bulkstats                       Enabled
```

```
Event notification level          Disabled
SNMP notifications                Disabled
REST interface authentication     peer-fail
REST interface certificate        rest-cert
REST interface host name          Not configured

Interface               Status            Port
-----------------------------------------------
NETCONF                 Enabled            123
REST                    Enabled            234
```

Bulkstats Configuration

See below for a sample bulkstats configuration:

```
 [local]<host_name># show config bulkstats
 config
    bulkstats collection
    bulkstats mode
       file 1
          schema common format %uptime%,%host%,%ipaddr%
       #exit
       file 2
          schema system format %uptime%,%host%,%ipaddr%
       #exit
    #exit
End
```

Force Bulkstats Collection

See below for a sample to force statistics to be collected and pushed to the operational database for ConfD:

```
[local]<host_name># bulkstats force gather
```

Notes:

- Statistics will generally be pushed per collection interval timer configured for bulkstats.

Using Curl to Read Statistics

See below for a sample use of curl to read statistics via the server ConfD RESTful interface:

```
[<user>@server] ]$ curl -u admin:pswd!
https://rtp-mitg-si06.cisco.com:234/api/operational/bulkstats-operational?deep --cert
/users/<user>/ssl_cert/client_cert/client.crt --key
/users/<user>/ssl_cert/client_cert/client.key --cacert
/users/<user>/ssl_cert/root_cert/rootCA.pem

<bulkstats-operational xmlns="http://www.cisco.com/staros-bulkstats"
xmlns:y="http://tail-f.com/ns/rest"
xmlns:staros_bulkstats="http://www.cisco.com/staros-bulkstats">
   <file>
      <number>1</number>
      <schemas>
         <schema>system</schema>
         <names>
           <name>common</name>
           <key_ids>
             <key_id>none</key_id>
             <variable>
               <name>host</name>
               <value><host_name></value>
             </variable>
             <variable>
                <name>ipaddr</name>
                <value>1.2.3.4</value>
             </variable>
             <variable>
                <name>uptime</name>
                <value>5781</value>
```

```
            </variable>
          </key_ids>
        </names>
      </schemas>
   </file>
   <file>
      <number>2</number>
      <schemas>
         <schema>system</schema>
         <names>
            <name>system</name>
            <key_ids>
               <key_id>none</key_id>
               <variable>
                  <name>host</name>
                  <value><host_name></value>
               </variable>
               <variable>
                  <name>ipaddr</name>
                  <value>1.2.3.4</value>
               </variable>
               <variable>
                  <name>uptime</name>
                  <value>5781</value>
               </variable>
            </key_ids>
         </names>
      </schemas>
   </file>
</bulkstats-operational>
```

Exec CLI Model

The following examples use the Exec CLI model.

Using Curl to Obtain the 'show version' Output

See below for a sample use of curl to obtain the **show version** output:

```
cat exec_cli_show_version.xml
<input><args>show version</args></input>
 ************
[<user>@server] ]$ curl -u admin:pswd!
https://rtp-mitg-si06.cisco.com:234/api/running/staros_exec/_operations/exec --cert
/users/<user>/ssl_cert/client_cert/client.crt --key
/users/<user>/ssl_cert/client_cert/client.key --cacert
/users/<user>/ssl_cert/root_cert/rootCA.pem -X POST -T ./exec_cli_show_version.xml
<output xmlns='http://www.cisco.com/staros-exec'>
   <result>Active Software:
   Image Version:          21.2.M0.private
   Image Build Number:     private
   Image Description:      Developer_Build
   Image Date:             Thu Feb 23 15:25:47 EST 2017
   Boot Image:             /flash/qvpc-si.bin.confd
   Source Commit ID:       bd234043a93c68873ea77444733a8c632356d161
 </result>
 </output>
```

Using Curl to Obtain Multiple Show Command Outputs

See below for a sample use of curl to obtain the **show build** and **show confdmgr**outputs, using "\r\n" as the delimiter between commands:

```
cat exec_cli_show_build_and_confdmgr.xml
<input><args>show build \r\n show confdmgr</args></input>
 ************
[<user>@server] ]$ curl -u admin:pswd!
https://rtp-mitg-si06.cisco.com:234/api/running/staros_exec/_operations/exec --cert
```

```
/users/<user>/ssl_cert/client_cert/client.crt --key
/users/<user>/ssl_cert/client_cert/client.key --cacert
/users/<user>/ssl_cert/root_cert/rootCA.pem -X POST -T ./ exec_cli_show_build_and_confdmgr.xml
<output xmlns='http://www.cisco.com/staros-exec'>
   <result>Active Software:
   Image Version:                  21.2.M0.private
   Image Build Number:             private
   Image Description:              Developer_Build
   Image Date:                     Thu Feb 23 15:25:47 EST 2017
   Boot Image:                     /flash/qvpc-si.bin.confd
   Source Commit ID:               bd234043a93c68873ea77444733a8c632356d161
   Kernel Version:                 2.6.38-staros-v3-ssi-64
   Kernel Machine Type:            x86_64

Build Information:
   Kernel Build:                   #1 SMP PREEMPT Wed Feb 22 12:28:49 EST 2017
   Image Build Type:               Production build
   Image Build User:               <user>
   Image Build Machine:            <host_name>
   Image Build Changeset Version:  +
   Image Build Changeset Author:   <user>
   Image Build Changeset Location: cisco.com
   Image Build Changeset Number:   bd234043a93c68873ea77444733a8c632356d161
   Image Build Changeset PID:      2017-02-23
*********************************
***** Local changes exist *****
*********************************

State Information
-----------------
State                           Started
Subscriptions                   5
Last successful id              1488-216669-170664
Last failed id                  None
Username                        Not configured
Bulkstats                       Enabled
Event notification level        Disabled
SNMP notifications              Disabled
REST interface authentication   peer-fail
REST interface certificate      rest-cert
REST interface host name        Not configured

Interface              Status           Port
-----------------------------------------------
NETCONF                Enabled           123
REST                   Enabled           234

Statistics
----------
Triggers                    1
Replays                     0
Notifications               27
Notification failures       0
Trigger failures            0
Replay failures             0
NETCONF notification failures 0
Unexpected failures         0
</result>
</output>
***********************************
```

CLI Based YANG Model for ECS Commands

In this release, the **cisco-staros-cli-config.yang** model supports a limited set of ECS (Enhanced Charging System) configuration commands via NSO.

On the southbound side, ConfD communicates with a StarOS process called via a set of APIs provided by the ConfD management agent. The ConfD 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 the **cisco-staros-cli-config** YANG model supported configuration objects must be applied to the CDB manually.

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.

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 CLI based 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>#
```

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.