Yuma netconfd Manual

YANG-Based Unified Modular Automation Tools

NETCONF Over SSH Server

Version 1.15-1

April 17, 2011

Table Of Contents

Yuma netconfd Manual

1 Preface

1.1 Legal Statements

1.2 Additional Resources

This document assumes you have successfully set up the software as described in the printed document:

Yuma Installation Guide

Other documentation includes:

Yuma Quickstart Guide

Yuma User Manual

Yuma yangcli Manual

Yuma yangdiff Manual

Yuma yangdump Manual

Yuma Developer Manual

To obtain additional support you may send email to this e-mail address

andy@netconfcentral.org

The SourceForge.net Support Page for Yuma can be found at this WEB page:

http://sourceforge.net/projects/yuma/support

There are several sources of free information and tools for use with YANG and/or NETCONF.

The following section lists the resources available at this time.

1.2.1 WEB Sites

- **Netconf Central**
  - http://www.netconfcentral.org/
  - Yuma Home Page
    - Free information on NETCONF and YANG, tutorials, on-line YANG module validation and documentation database
- **Yuma SourceFource OpenSource Project**
  - http://sourceforge.net/projects/yuma/
    - Download Yuma source and binaries; project forums and help

- **Yang Central**
  - http://www.yang-central.org
  - Free information and tutorials on YANG, free YANG tools for download
- **NETCONF Working Group Wiki Page**
  - http://trac.tools.ietf.org/wg/netconf/trac/wiki
  - Free information on NETCONF standardization activities and NETCONF implementations
- **NETCONF WG Status Page**
  - http://tools.ietf.org/wg/netconf/
  - IETF Internet draft status for NETCONF documents
- **libsmi Home Page**
  - http://www.ibr.cs.tu-bs.de/projects/libsmi/
  - Free tools such as smidump, to convert SMIv2 to YANG

1.2.2 Mailing Lists

- **NETCONF Working Group**
  - http://www.ietf.org/html.charters/netconf-charter.html
  - Technical issues related to the NETCONF protocol are discussed on the NETCONF WG mailing list. Refer to the instructions on the WEB page for joining the mailing list.
- **NETMOD Working Group**
  - http://www.ietf.org/html.charters/netmod-charter.html
  - Technical issues related to the YANG language and YANG data types are discussed on the NETMOD WG mailing list. Refer to the instructions on the WEB page for joining the mailing list.

1.3 Conventions Used in this Document

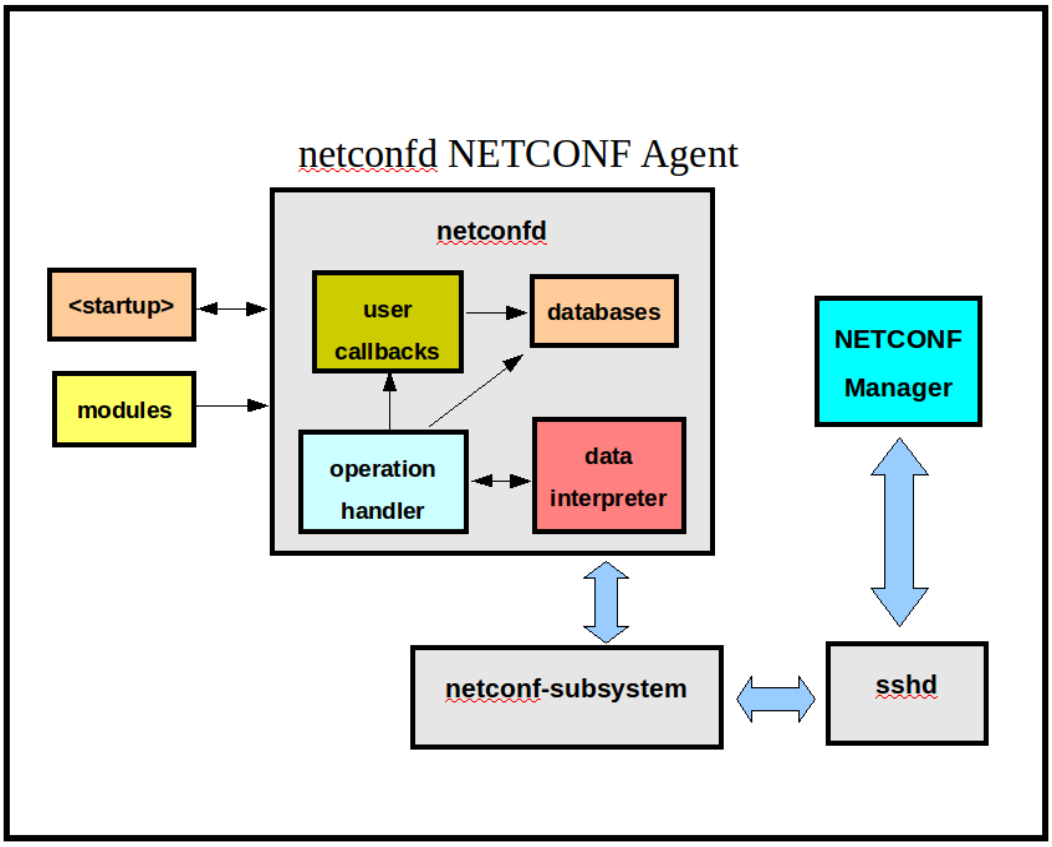
The following formatting conventions are used throughout this document:

Documentation Conventions

Convention	Description
--foo	CLI parameter foo
<foo>	XML parameter foo
foo	**yangcli** command or parameter
$$FOO	Environment variable FOO
$$foo	**yangcli** global variable foo
`some text`	Example command or PDU
some text	Plain text

2 netconfd User Guide

netconfd Program Components



2.1 Introduction

The **netconfd** program is a NETCONF-over-SSH server implementation. It is driven directly by YANG files, and provides a robust and secure database interface using standard NETCONF protocol operations.

All aspects of NETCONF protocol operation handling can be done automatically by the **netconfd** server. However, the interface between the NETCONF database and the device instrumentation is not covered in this document. Refer to the server Developers Guide for details on adding YANG module instrumentation code to the **netconfd** server.

2.1.1 Features

The **netconfd** server has the following features:

- Automatic support for all NETCONF operations, including the YANG 'insert' operation.
- Supports <candidate>, <running>, and <startup> databases

- Supports the complete NETCONF protocol defined in RFC 4741, and most of the new RFC 4741bis draft in progress in the IETF.
- Full, automatic run-time support for any YANG-defined NETCONF content:
  - rpc statement automatically supported, so new operations can be added at run-time
  - all YANG data statements automatically supported, so new database objects can be added at run-time
  - notification statement automatically supported, so new notification event types can be added at run-time
- Complete XML 1.0 implementation with full support for XML Namespaces
- Automatic support for all capability registration and <hello> message processing
- Full, automatic generation of all YANG module <capability> contents, including features and deviations
- Automatic session management, including unlimited number of concurrent sessions, session customization, and all database cleanup
- Full support for database locking, editing, validation, including extensive user-callback capabilities to support device instrumentation.
- Automatic support for all YANG validation mechanisms, including all XPath conditionals
- Automatic subtree and full XPath filtering
- Automatic confirmed-commit and rollback procedures
- Automatic database audit log and change notification support
- Complete <rpc-error> reporting support, including user-defined errors via YANG error-app-tag and error-message statements.
- Several <rpc-error> extensions, including <bad-value> and <error-number>, for easier debugging
- Comprehensive, fully NETCONF configurable, access control model, defined in **yuma-nacm.yang**.
- Complete RFC 5277 Notification support, including notification replay, :interleave capability, and several useful notifications implemented in **yuma-system.yang**.
- Complete RFC 5717 Partial Lock support with full XPath support, and all partial locking monitoring data defined in i**etf-netconf-monitoring.yang.**
- Full support for all YANG constructs, including deviations
- Full support of YANG sub-modules, including nested sub-modules
- Multiple concurrent module versions supported (import-by-revision)
- Multiple concurrent submodule versions supported (include-by-revision)
- Optimized, full XPath 1.0 implementation, including all YANG extensions
- Full implementation of the **ietf-netconf-monitoring** data model, including the <get-schema> operation to retrieve YANG modules from the server.
- Configurable default node handling, including full support of the <with-defaults> standard, in **ietf-with-defaults.yang**.
- System information automatically supported, as defined in **yuma-system.yang.**
- Comprehensive logging capabilities for easy debugging during YANG content development or normal operation.

2.1.2 Setting the Server Profile

The **netconfd** server can behave in different ways, depending on the initial configuration parameters used.

The following parameters should be considered, and if the default behavior is not desired, then an explicit value should be provided instead:

- **--yuma-home** or **$YUMA_HOME** setting will affect YANG search path.
- **--modpath** or **$YUMA_MODPATH** setting will affect YANG search path.
- **--datapath** or **$YUMA_DATAPATH** setting will affect startup-cfg.xml search path.
- **--target** setting will select the edit target. The default is 'candidate', so this parameter must be set to choose 'running' as the edit target.
- **--with-startup** setting will enable the <startup> database if set to 'true'.
- **--with-validate** setting will enable the :validate capability if set to 'true'
- **--access-control** setting will affect how access control is enforced. The default is fully on ('enforcing').
- **--superuser** setting will affect access control, if it is enabled. The default is 'superuser'.
- **--default-style** setting will affect how default leaf values are returned in retrieval requests. The default is 'trim'. This returns everything except leafs containing the YANG default-stmt value, by default. To report every leaf value by default, set this parameter to 'report-all'. To report only leafs not set by the server by default, use the 'explicit' enumeration.
- **--log** and **--log-level** settings will affect how log messages are generated. The default is to send all messages to STDOUT, and use the 'info' logging level. If the server is a DEBUG image, then the default logging level will be 'debug' instead.
- **--eventlog-size** setting will control the memory used by the notification replay buffer
- **--max-burst** will control the of notifications sent at once to a single session
- **--hello-timeout** will control how long sessions can be stuck waiting for a hello message before they are dropped.
- **-- idle-timeout** will control how long active sessions can remain idle before they are dropped.

2.1.3 Loading YANG Modules

The --**module** parameter can be used from the CLI or .conf file to pre-load YANG modules and any related device instrumentation code into the server. A fatal error will occur if any module cannot be loaded, or it contains any YANG errors.

At run-time, the <load> operation (defined in **yuma-system.yang**) can be used to do the same thing, except the server will simply return an <rpc-error> instead of terminate, if the requested module cannot be loaded.

2.1.4 Starting netconfd

The current working directory in use when **netconfd** is invoked is important. It is most convenient to run **netconfd** in the background, and save all output to a log file.

The **netconfd** program listens for connection requests on a local socket, that is located in **/tmp/ncxserver.sock**.

In order for NETCONF sessions to be enabled, the **SSH server** and the **netconf-subsystem** programs must be properly installed first.

The **netconfd** program does not directly process the SSH protocol messages. Instead, it is implemented as an SSH subsystem. The number of concurrent SSH sessions that can connect to the **netconfd** server at once may be limited by the SSH server configuration. Refer to the Yuma Installation Guide for more details on configuring the SSH server for use with **netconfd**.

The **netconfd** program can be invoked several ways:

- To get the current version and exit:

```
netconfd --version
```

- To get program help and exit:

```
netconfd --help
netconfd --help --brief
netconfd --help --full
```

- To start the server in the background, set the loggin level to 'debug', and send logging messages to a log file

```
netconfd --log-level=debug --log=~/mylog &
```

- To start the server interactively, and send all log messages to STDOUT:

```
netconfd
```

- To start the server interactively, with a new log file:

```
netconfd --logfile=mylogfile
```

- To start the server interactively, and append to an existing log file:

```
netconfd --logfile=mylogfile --log-append
```

- To get parameters from a configuration file:

```
netconfd --config=/etc/yuma/netconfd.conf
```

2.1.5 Stopping netconfd

To terminate the **netconfd** program when running interactively, use the control-C character sequence. This will cause the server to cleanup and terminate gracefully.

The <shutdown> or <restart> operations can also be used to terminate or restart the server. The **yuma-nacm.yang** access control rules must be configured to allow any user except the 'superuser' account to invoke this operation.

2.1.6 Signal Handling

The server will respond to Unix signals sent to the **netconfd** process.

If the server is being run in the foreground, then the Control-C character sequence will perform the same action as a SIGINT signal.

Signals Recognized by netconfd

signal	number	description
SIGHUP (Hangup)	1	Restart the server.
SIGINT (Control-C)	2	Shutdown the server.
SIGQUIT	3	Shutdown the server.
SIGILL	4	Shutdown the server.
SIGTRAP	5	Shutdown the server.
SIGABRT	6	Shutdown the server.
SIGKILL	9	Shutdown the server.
SIGPIPE	13	Handle I/O connection error.
SIGTERM	15	Shutdown the server.

The **kill** command in Unix can be used to send signals to a process running in the background. Refer to the Unix man pages for more details.

2.1.7 Error Handling

All of the error handling requirements specified by the NETCONF protocol, and the YANG language error extensions for NETCONF, are supported automatically by **netconfd**.

There are 4 categories of error handling done by the server:

- incoming PDU validation

    - Errors for invalid PDU contents are reported immediately. The server will attempt to find all the errors in the input <rpc> request, and not stop detecting errors after one is found.
    - All machine-readable YANG statements are utilized to automate the detection and reporting of errors.
    - A node that is present, but has a false 'when' statement, is treated as an error.
- server instrumentation PDU validation
    - Semantic requirements expressed only in description statements will be checked by device instrumentation callbacks. The specific YANG data module should indicate which errors may be reported, and when they should be reported.
- database validation
    - Several automated tests are performed when a database is validated.
    - If the edit target is the <candidate> configuration, then referential integrity tests are postponed until the <commit> operation is attempted.
    - The specific conditions checked automatically are:
        - referential integrity condition test failed (must)
        - missing leaf (mandatory)
        - missing choice (mandatory)
        - extra container or leaf
        - too few instances of a list or leaf-list (min-elements)
        - too many instances of a list or leaf-list (max-elements)
        - instance not unique (unique)
    - Nodes that are unsupported by the server will automatically be removed from these tests. This can occur in the following ways:
        - node is defined within a feature that is not supported (if-feature)
        - node has conditional existence test that is false (when)
        - nodes derived from a 'uses' statement which has a conditional existence test that is false (when)
        - nodes derived from an 'augment' statement which has a conditional existence test that is false (when)
- server instrumentation database validation and activation
    - Errors can occur related to the specific YANG data model module, which can only be detected and reported by the server instrumentation.
    - Resource denied errors can occur while the server instrumentation is attempting to activate the networking features associated with some configuration parameters.
    - Instrumentation code can fail for a number of reasons, such as underlying hardware failure or removal.

2.1.8 Module Summary

The following YANG modules are built into the netconfd server, and cannot be loaded manually with the **module** parameter or <load> operation.

Pre-loaded YANG Modules

module	description
ietf-inet-types	standard data types
ietf-netconf-monitoring	standard NETCONF monitoring, and the <get-schema> operation
ietf-netconf-partial-lock	standard NETCONF <partial-lock> and <partial-unlock> operations
ietf-with-defaults	<with-defaults> extension
ietf-yang-types	standard data types
yuma-interfaces	network interfaces information
yuma-nacm	NETCONF Access Control Model
nc-notifications	standard replay notifications
netconfd	Server CLI parameters
notifications	standard notification operations
yuma-ncx	Yuma NETCONF extensions
yuma-app-common	Common CLI parameters
yuma-types	Yuma common data types
yuma-mysession	Get and Set session-specific parameters
yuma-proc	/proc file system monitoriing
yuma-system	system monitoring, operations, and notifications

2.1.9 Notification Summary

The following notification event types are built into the **netconfd** server:

Pre-loaded Notifications

module	event type	description
nc-notifications	<replayComplete>	Notification replay has ended
nc-notifications	<notificationComplete>	Notification delivery has ended
yuma-system	<sysStartup>	server startup event
yuma-system	<sysSessionStart>	NETCONF session started
yuma-system	<sysSessionEnd>	NETCONF session ended
yuma-system	<sysConfigChange>	<running> configuration has changed
yuma-system	<sysCapabilityChange>	server capability added or deleted
yuma-system	<sysConfirmedCommit>	confirmed-commit procedure event

2.1.10 Operation Summary

The following protocol operations are built into the **netconfd** server:

Pre-loaded Operations

module	operation	description
ietf-netconf	<close-session>	Terminate the current session.
ietf-netconf	<commit>	Activate edits in <candidate>.
ietf-netconf	<copy-config>	Copy an entire configuration.
notifications	<create-subscription>	Start receiving notifications.
ietf-netconf	<delete-config>	Delete a configuration.
ietf-netconf	<discard-changes>	Discard edits in <candidate>.
ietf-netconf	<edit-config>	Edit the target configuration.
ietf-netconf	<get>	Retrieve <running> or state data.
ietf-netconf	<get-config>	Retrieve all or part of a configuration.
yuma-mysession	<get-my-session>	Retrieve session customization parameters.
ietf-netconf-monitoring	<get-schema>	Retrieve a YANG module definition file.
ietf-netconf	<kill-session>	Terminate a NETCONF session.
netconfd	<load>	Load a YANG module.
ietf-netconf	<lock>	Lock a database.
netconfd	<no-op>	No operation.
ietf-netconf-partial-lock	<partial-lock>	Lock part of the <running> database
ietf-netconf-partial-lock	<partial-unlock>	Unlock part of the <running> database
netconfd	<restart>	Restart the server.
yuma-mysession	<set-my-session>	Set the session customization parameters.
yuma-system	<set-log-level>	Set the logging verbosity level.
netconfd	<shutdown>	Shutdown the server.
ietf-netconf	<unlock>	Unlock a database.
ietf-netconf	<validate>	Validate a database

2.1.11 Configuration Parameter List

The following configuration parameters are used by **netconfd**. Refer to the CLI Reference for more details.

netconfd CLI Parameters

parameter	description
--access-control	Specifies how access control will be enforced
--config	Specifies the configuration file to use for parameters
--datapath	Specifies the search path for the <startup> configuration file.
--default-style	Specifies the default <with-defaults> behavior
--deviation	Species one or more YANG modules to load as deviations
--eventlog-size	Specifies the maximum number of events stored in the notification replay buffer.
--feature-disable	Leaf list of features to disable
--feature-enable	Specifies a feature that should be enabled
--feature-enable-default	Specifies if a feature should be enabled or disabled by default
--help	Get context-sensitive help with --brief or --full extension
--hello-timeout	Set the number of seconds to wait for a <hello> PDU
--idle-timeout	Set the number of seconds to wait for a <rpc> PDU
--indent	Specifies the indent count to use when writing data
--log	Specifies the log file to use instead of STDOUT
--log-append	Controls whether a log file will be reused or overwritten
--log-level	Controls the verbosity of logging messages
--max-burst	Specifies the maximum number of notifications to send to one session in a row.
--modpath	Sets the module search path
--module	Specifies one or more YANG modules to load upon startup
--no-startup	If present, the startup configuration will not be used (if present), and the factory defaults will be used instead.
--port	Specifies up to 4 TCP port numbers to accept NETCONF connections from
--runpath	Script search path (for instrumentation use only at this time)
--startup	Specifies the startup configuration file location to override the default. Not allowed if the --no-startup parameter is present.
--startup-error	Specifies whether the server should stop or continue if the startup configuration contains any errors.
--subdirs	If true, then sub-directories will be searched when

	looking for files. Otherwise just the specified directory will be used and none of its sub-directories (if any).
--superuser	Specifies the user name (or empty string for none) to be given super user privileges, instead of the default 'superuser'.
--target	Specifies if the <candidate> or <running> configuration should be the edit target
--usexmlorder	Forces strict YANG XML ordering to be enforced
--version	Prints the program version and exits
--warn-idlen	Controls how identifier lengths are checked
--warn-linelen	Controls how line lengths are checked
--warn-off	Suppresses the specified warning number
--with-startup	Enable or disable the <startup> database
--with-url	Enable or disable the **:url** capability
--with-validate	Enable or disable the **:validate** capability
--yuma-home	Specifies the **$YUMA_HOME** project root to use when searching for files

2.2 Capabilities

Server capabilities are the primary mechanism to specify optional behavior in the NETCONF protocol. This section describes the capabilities that are supported by the **netconfd** server.

2.2.1 :candidate

The **:candidate** capability indicates that database edits will be done to the <candidate> database, and saved with the <commit> operation.

The <candidate> configuration is a shared scratch-pad, so it should be used with the locking. Database edits are collected in the <candidate> and then applied, all or nothing, to the <running> database, with the <commit> operation.

This capability is supported by default. It is controlled by the --**target** configuration parameter (--target=candidate).

By default, only the superuser account can use the <delete-config> operation on the <candidate> configuration.

2.2.2 :confirmed-commit

The **:confirmed-commit** capability indicates that the server will support the <confirmed> and <confirm-timeout> parameters to the <commit> operation.

The confirmed commit procedure requires that two <commit> operations be used to apply changes from the candidate configuration to the running configuration.

If the second <commit> operation is not received before the <confirm-timeout> value (default 10 minutes), then the running and candidate configurations will be reset to the contents of the running configuration, before the first <commit> operation.

If the session that started the confirmed-commit procedure is terminated for any reason before the second <commit> operation is completed, then the running configuration will be reset, as if the confirm-timeout interval had expired.

If the confirmed-commit procedure is used, and the :startup capability is also supported, then the contents of NV-storage (e.g., startup-cfg.xml) will not be updated or altered by this procedure. Only the running configuration will be affected by the rollback,

If the <confirmed> parameter is used again, in the second <commit> operation, then the timeout interval will be extended, and any changes made to the candidate configuration will be committed. If the running and candidate configurations are reverted, any intermediate edits made since the first <commit> operation will be lost.

2.2.3 :interleave

The **:interleave** capability indicates that the server will accept <rpc> requests other than <close-session> during notification delivery. It is supported at all times, and cannot be configured.

2.2.4 :netconf-monitoring

The **:netconf-monitoring** capability indicates that the **/ietf-netconf-monitoring** data sub-tree is supported.

The netconfd server supports all of the tables in this module, except partial-locking, because the **:partial-lock** capability is not supported at this time.

The **/netconf-state/capabilities** subtree can be examined to discover the active set of NETCONF capabilities.

The **/netconf-state/datastores** subtree can be examined to discover the active database locks.

The **/netconf-state/schemas** subtree can be examined for all the YANG modules that are available for download with the <get-schema> operation.

The **/netconf-state/sessions** subtree can be examined for monitoring NETCONF session activity.

The **/netconf-state/statistics** subtree can be examined for monitoring global NETCONF counters.

2.2.5 :notification

The **:notification** capability indicates that the server will accept the <create-subscription> operation, and deliver notifications to the session, according to the subscription request.

All <create-subscription> options and features are supported.

A notification log is maintained on the server, which is restarted every time the server reboots.

This log can be accessed as a 'replay subscription'.

The first notification in the log will be for the <sysStartup> event.

The <replayComplete> and <notificationComplete> event types are not stored in the log.

2.2.6 :partial-lock

The **:partial-lock** capability indicates that RFC 5717 is implemented, and partial locking of the <running> database is supported.

The <copy-config> operation is not supported using the <running> database as a target, so partial locks do not affect that operation.

The <edit-config> operation on the <running> database is allowed if the --target parameter is set to 'running'.

The <commit> operation will fail if any portion of the altered configuration is locked by another session. Data in the <candidate> database which is identical to the corresponding data in the <running> configuration is not affected by a <partial-lock> operation.

The constant **VAL_MAX_PLOCKS** in ncx/val.h controls the maximum number of concurrent locks that a single session can own on a database node. The default value is 4.

There is no hard resource limit for:

- the number of total partial-locks
- the number of <select> parameters in the <partial-lock> request
- the number of nodes that can be locked by a single partial-lock

When the maximum <lock-id> is reached (MAX_UINT), the server will not reset the <lock-id> to '1' unless there are no partial locks currently held on the <running> database. The <lock-id> '0' is not used.

2.2.7 :rollback-on-error

The **:rollback-on-error** capability indicates that the server supports 'all-or-nothing' editing for a single <edit-config> operation. This is a standard enumeration value for the <error-option> parameter.

The server will perform all PDU validation no matter what <error-option> is selected.

Execution phase will not occur if any errors at all are found in the validation phase.

During execution phase, this parameter will affect error processing. When set to rollback-on-error, if any part of the requested configuration change cannot be performed, the database will be restored to its previous state, and server instrumentation callbacks to 'undo' any changes made will be invoked.

2.2.8 :schema-retrieval

The **:schema-retrieval** capability indicates that the <get-schema> operation is supported.

The **netconfd** server supports this operation for all YANG modules in use at the time.

The <identifier> parameter must be set to the name of the YANG module.

The <format> parameter must be set to 'YANG'.

The <version> parameter must be set to a revision date to retrieve a specific version of the module, or the empty string to retrieve whatever version the server is using.

2.2.9 :startup

The **:startup** capability indicates that the <startup> configuration is supported by the server.

By default, this capability is not supported.

This capability is controlled by the **--with-startup** configuration parameter. If this parameter is set to 'true', then the :startup capability will be supported.

If this capability is supported:

- the server will allow the <startup> configuration to be the source of a <get-config>.
- the server will allow the <startup> configuration to be the target of a <copy-config> operation, if the source is the <running> configuration.

- If the :validate capability is enabled, then the server will allow the <startup> configuration to be the target of a <validate> operation.
- If the user is the super user account, or access is configured in NACM to allow it, then the server will allow the <startup> configuration to be the target of a <delete-config> operation.

No other operations on the <startup> database are supported. The <startup> database cannot be edited with <edit-config>, or over-written with <copy-config>.

2.2.10 :VALIDATE

The **:validate** capability indicates that the <validate> operation is accepted, and the <test-option> for the <edit-config> operation is also accepted, by the server.

This capability is controlled by the **--with-validate** configuration parameter. If it is set to 'false' then this capability will not be available in **netconfd**.

The <validate> operation can be invoked in several ways:

- validate the <candidate> database if the **:candidate** capability is supported
- validate the <running> database
- validate the <startup> database if the **:startup** capability is supported
- validate an inline <config> element, which represents the entire contents of a database.

The <test-option> parameter for the <edit-config> operation can be used. This parameter has a significant impact on operations, and needs to be used carefully.

- **set**: This option is not the default, but it is probably the desired behavior for the <candidate> database. In order to fully utilize the incremental editing capability of this database, the 'set' value should be used. This will prevent any validation error messages unrelated to the current edit. The <validate> operation can be used before the <commit> is done, if desired. The same errors (if any) should be reported by <validate> or <commit>.
- **test-then-set**: This option is the default, and will cause a resource intensive validation procedure to be invoked every time the <candidate> database is edited. (The validation procedure is always invoked on every edit to the <running> configuration.) If any database validation errors are found (in addition to the requested edit), then they will be reported. Use the <error-path> field to determine which type of error is being reported by the server.

2.2.11 :URL

The **:url** capability indicates that the server accepts the <url> parameter in NETCONF operations that use this parameter. This capability can be disabled with the **--with-url** CLI configuration parameter.

The following operations are affected by the :url capability:

- edit-config
  - <url> accepted instead of <config> as a configuration data source
- copy-config
  - <url> accepted as a source parameter.
  - <url> accepted as a target paramter.
  - <url> to <url> copy not supported (optional to implement)
- delete-config:
  - <url> accepted as source to delete

- validate
  - <url> accepted as a configuration source to validate.

Only the 'file' scheme is supported at this time.

A URL file can be specified as a simple file within the root directory.

No whitespace or special characters are allowed in the file name.

The file extension '.xml' should be used. The server only generates and expects XML configuration files. The NETCONF 'config' element is used as the top-level element in all <url> files.

The $YUMA_DATAPATH environment variable or the $$datapth system variable is used to find the file names specified in the <url> URI string.

Example:

```
      <url>file:///my-backup.xml</url>
```

2.2.12 :WITH-DEFAULTS

The **:with-defaults** capability indicates that the server will accept the <with-defaults> parameter for the following operations:

- <get>
- <get-config>
- <copy-config>

There are 3 values defined for this parameter. The server supports all of them, no matter what mode is used for the default style.

- **report-all**: all nodes are reported
- **trim**: nodes set to thier YANG default-stmt value by the server or the client are skipped
- **explicit**: nodes set by the server are skipped. (Nodes set in the <startup> are considered to be set by the client, not the server.)

The **--default-style** configuration parameter is used to control the behavior the server will use for these operations when the <with-defaults> parameter is missing. The server will also use this default value when automatically saving the <running> configuration to non-volatile storage.

2.2.13 :WRITABLE-RUNNING

The **:writable-running** capability indicates that the server uses the <running> configuration as its edit target. In this case, the <target> parameter for the <edit-config> operation can only be set to <running/>.

All edits are activated immediately, but only if the entire database is going to be valid after the edits. A non-destructive test is performed before activating the requested changes.

If this capability is advertised, then **netconfd** will also advertise the :startup capability. They are always used together.

Edits to the <running> configuration take affect right away, but they are only saved to non-volatile storage automatically if the **with-startup** configuration parameter is set to 'false'.

2.2.14 :XPATH

The :xpath capability indicates that XPath filtering is supported for the <get> and <get-config> operations.

The netconfd server implements all of XPath 1.0, plus the following additions:

- the current() function from XPath 2.0 is supported
- a missing XML namespace will match any YANG module namespace (XPath 2.0 behavior) instead of matching the NULL namespace (XPath 1.0 behavior)
- the 'preceding' and 'following' axes should not be used. The database is dynamic and the relative document order is not stable. This is also a very resource intensive operation.

XPath filtering affects whether the server will return particular subtrees or not. It does not change the format of the <get> or <get-config> output. The result returned by the server will not be the raw XPath node-set from evaluating the specified 'select' expression against a database.

The server will normalize the XPath search results it returns:

- There will be no duplicate nodes, even if there were duplicates in the XPath result node-set.
- All result nodes with common ancestor nodes will be grouped together in the <rpc-reply>.
- All list nodes will include child nodes for any missing key leafs, even if they were not requested in the 'select' expression.

2.3 Databases

A NETCONF database is conceptually represented as an XML instance document.

There are 3 conceptual databases, which all share the exact same structure.

- <candidate>: scratch-pad to gather edits to apply to <running> all at once
- <running>: configuration data in effect
- <startup>: configuration data for the next reboot

When the <running> configuration is saved to non-volatile storage, the top-level element of this document is the <config> container element.

The XML namespace of this element is the **netconfd** module namespace, but a client application should expect that other server implementations may use a different namespace, such as the NETCONF namespace, or perhaps no namespace at all for this top-level element.

When database contents are returned in the <get>, <get-config>, or <copy-config> operations, the top-level container will be the <data> element in the NETCONF base namespace.

The top-level YANG module data structures that are present in the configuration will be present as child nodes of the <config> or <data> container node.

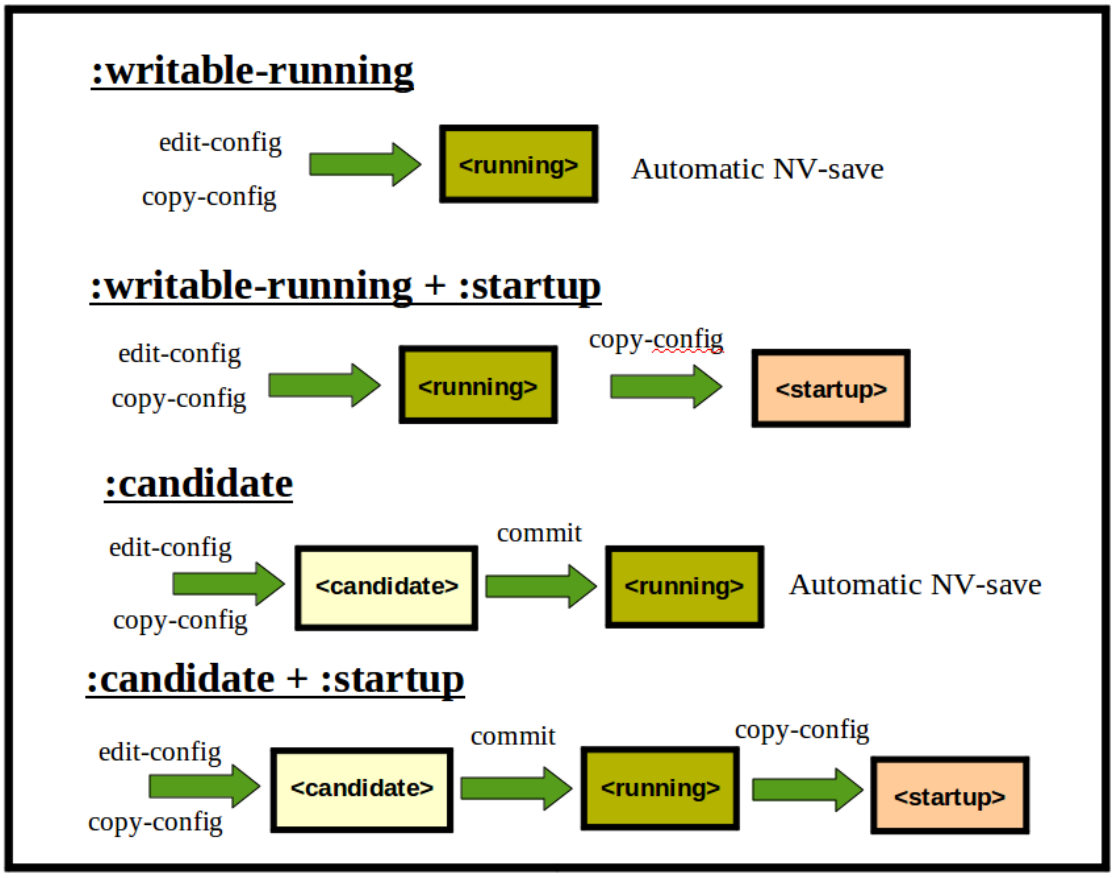
The exact databases that are present in the server are controlled by 3 capabilities:

- :candidate
- :writable-running
- :startup

The edit target in the server is set with the **--target** configuration parameter. This will select either the **:candidate** or **:writable-running** capabilities.

The server behavior for non-volatile storage of the <running> configuration is set with the **--with-startup** configuration parameter. The **:startup** capability will be supported if this parameter is set to 'true'.

The following diagram shows the 4 database usage modes that **netconfd** supports:



2.3.1 Database Locking

It is strongly suggested that the <lock> and <unlock> operations be used whenever a database is being edited. All the databases on the server should be locked, not just one, because different operations are controlled by different locks. The only way to insure that the entire database transaction is done in isolation is to keep all the databases locked during the entire transaction.

The affected configurations should be locked during the entire transaction, and not released until the edits have been saved in non-volatile storage.

If the edit target is the <candidate> configuration, then the <candidate> and <running> configurations should be locked.

If the edit target is the <running> configuration, then the <running> and <startup> configurations should be locked.

Whenever the lock on the <candidate> configuration is released, a <discard-changes> operation is performed by the server. This is required by the NETCONF protocol.

Of the <candidate> configuration contains any edits, then a lock will fail with a 'resource-denied' error. In this case, a lock on the <candidate> configuration cannot be granted until the <discard-changes> operation is completed.

2.3.2 Using the <candidate> Database

The <candidate> database is available if the **:candidate** capability is advertised by the server.

The <lock> operation will fail on this database if there are any edits already pending in the <candidate>. If a 'lock-failed' error occurs and no session is holding a lock, then use the <discard-changes> operation to clear the <candidate> buffer of any edits.

Once all the edits have been made, the <validate> operation can be used to check if all database validation tests will pass. This step is optional.

Once the edits are completed, the <commit> operation is used to activate the configuration changes, and save them in non-volatile storage.

The <discard-changes> operation is used to clear any edits from this database.

2.3.3 Using the <running> Database

The <running> database is available at all times for reading.

If the :writable-running capability is advertised by the server, then this database will be available as the target for <edit-config> operations.

Edits to the <running> configuration will take affect right away, as each <edit-config> operation is completed.

Once all the edits are completed, the <copy-config> operation can be used to save the current <running> configuration to non-volatile storage, by setting the target of the <copy-config> operation to the <startup> configuration.

2.3.4 Using the <startup> Database

The <startup> database is available if the :startup capability is advertised by the server.

The <copy-config> operation can be used to save the contents of the <running> configuration to the <startup> configuration.

The <get-config> operation can be used to retrieve the contents of the <startup> configuration.

The <delete-config> operation can be used to delete the <startup> configuration. Only the superuser account is allowed to do this, by default.

2.4 Sessions

All NETCONF server access is done through the NETCONF protocol, except the server can be shutdown with the Control-C character sequence if it being run interactively.

This section describes any **netconfd** implementation details which may NETCONF sessions.

2.4.1 User Names

The user name string associated with a NETCONF session is derived from the **$SSH_CONNECTION** environment variable, which is available to the **netconf-subsystem** program when it is called by **sshd**.

Any user name accepted by **sshd** will be accepted by **netconfd**.

In order for access control to work properly, the **sshd** user name must also conform to the **NacmUserName** type definition:

```
typedef NacmUserName {
    description "General Purpose User Name string.";
    type string {
       length "1..63";
       pattern '[a-z,A-Z][a-z,A-Z,0-9]{0,62}';
    }
}
```

- A user name cane be 1 to 63 characters long.
- The first character must be a letter ['a' to 'z', or 'A' to 'Z']
- The remaining characters must be a letter ['a' to 'z', or 'A' to 'Z'], or a number ['0' to '9'].

2.4.2 Session ID

The <session-id> assigned by the server is simply a monotonically increasing number:

```
typedef SessionId {
    description "NETCONF Session Id";
    type uint32 {
       range "1..max";
    }
}
```

The server will start using session ID values over again at 1, if the maximum session-id value is ever reached.

2.4.3 Server <hello> Message

The **netconfd** server will send a <hello> message if a valid SSH2 session to the netconf subsystem is established.

The server will list all the capabilities it supports.

The YANG module capability URI format is supported for all modules, including ones that only contain typedefs or groupings.

The URI format is defined in the YANG specification, and follows this format:

<module-namespace> '?module='<module-name>'&revision='<module-date>'

If the module does not have any revision statements, then the revision field will not be present in the module capability URI.

If the module contains any supported features, then the following field will be added, and each supported feature name will be listed:

'&features='<feature-name> [','<feature-name>]*

If the module needs any external deviations applied, then the following field will be added, and each deviation module name will be listed:

'&deviations='<deviation-module-name> [','<deviation-module-name>]*

Note that the deviation modules will be listed in the capabilities, along with other modules. The 'deviations' extension allows a client tool to know that the deviations apply to the specific module, since special processing may be required.

Example server <hello> Message:

```
<nc:hello xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0">
   <nc:capabilities>
      <nc:capability>urn:ietf:params:netconf:base:1.0</nc:capability
>
      <nc:capability>
         urn:ietf:params:netconf:capability:candidate:1.0
      </nc:capability>
      <nc:capability>
         urn:ietf:params:netconf:capability:confirmed-commit:1.0
      </nc:capability>
      <nc:capability>
         urn:ietf:params:netconf:capability:rollback-on-error:1.0
      </nc:capability>
      <nc:capability>
         urn:ietf:params:netconf:capability:validate:1.0
      </nc:capability>
      <nc:capability>
         urn:ietf:params:netconf:capability:xpath:1.0
      </nc:capability>
      <nc:capability>
         urn:ietf:params:netconf:capability:notification:1.0
      </nc:capability>
      <nc:capability>
         urn:ietf:params:netconf:capability:interleave:1.0
      </nc:capability>
      <nc:capability>
         urn:ietf:params:netconf:capability:with-defaults:1.0?
basic=explicit&amp;supported=report-all:trim
      </nc:capability>
      <nc:capability>
         urn:ietf:params:netconf:capability:netconf-monitoring:1.0
      </nc:capability>
      <nc:capability>
         urn:ietf:params:netconf:capability:schema-retrieval:1.0
      </nc:capability>
      <nc:capability>
         urn:ietf:params:xml:ns:yang:inet-types?module=ietf-inet-
types&amp;revision=2009-05-13
      </nc:capability>
      <nc:capability>
         urn:ietf:params:xml:ns:netconf:monitoring?module=ietf-
netconf-monitoring&amp;revision=2009-06-16
      </nc:capability>
```

```
      <nc:capability>
         urn:ietf:params:netconf:capability:with-defaults:1.0?
module=ietf-with-defaults&amp;revision=2009-07-01&amp;features=with-
defaults
      </nc:capability>
      <nc:capability>
         urn:ietf:params:xml:ns:yang:yang-types?module=ietf-yang-
types&amp;revision=2009-05-13
      </nc:capability>
      <nc:capability>
         http://netconfcentral.org/ns/yuma-interfaces?
module=interfaces&amp;rQ

ses_msg: setup send buff 1
evision=2009-07-17
      </nc:capability>
      <nc:capability>
         http://netconfcentral.org/ns/yuma-mysession?module=yuma-
mysession&amp;revision=2009-08-11
      </nc:capability>
      <nc:capability>
         http://netconfcentral.org/ns/yuma-nacm?module=yuma-
nacm&amp;revision=2009-05-13
      </nc:capability>
      <nc:capability>
         urn:ietf:params:xml:ns:netmod:notification?module=nc-
notifications&amp;revision=2008-07-14
      </nc:capability>
      <nc:capability>
         http://netconfcentral.org/ns/yuma-ncx?module=yuma-
ncx&amp;revision=2009-06-12
      </nc:capability>
      <nc:capability>
         http://netconfcentral.org/ns/yuma-app-common?module=yuma-
app-common&amp;revision=2009-04-10
      </nc:capability>
      <nc:capability>
         http://netconfcentral.org/ns/yuma-types?module=yuma-
types&amp;revision=2008-07-20
      </nc:capability>
      <nc:capability>
         http://netconfcentral.org/ns/netconfd?
module=netconfd&amp;revision=2009-05-28
      </nc:capability>
      <nc:capability>
         urn:ietf:params:xml:ns:netconf:notification:1.0?
module=notifications&amp;revision=2008-07-14
      </nc:capability>
      <nc:capability>
         http://netconfcentral.org/ns/yuma-proc?module=yuma-
proc&amp;revision=2009-07-17
      </nc:capability>
      <nc:capability>
         http://netconfcentral.org/ns/yuma-system?module=yuma-
system&amp;revision=2009-06-04
      </nc:capability>
      <nc:capability>
         http://netconfcentral.org/ns/test?
module=test&amp;revision=2009-06-
10&amp;features=feature1,feature3,feature4
      </nc:capability>
   </nc:capabilities>
   <nc:session-id>1</nc:session-id>
```

```
</nc:hello>]]>]]>
```

2.4.4 Client <hello> Message

The **netconfd** server requires a valid <hello> message from the client before accepting any <rpc> requests.

Only the mandatory 'netconf base' URI will be checked by the server. All other <capability> elements will be ignored by the server.

Example client <hello> Message:

```
<?xml version="1.0" encoding="UTF-8"?>
<nc:hello xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0">
   <nc:capabilities>
      <nc:capability>urn:ietf:params:netconf:base:1.0</nc:capability
>
   </nc:capabilities>
</nc:hello>
```

2.4.5 RPC Request Processing

The only PDU the netconfd server will accept during a NETCONF session is the <rpc> message.

All aspects of NETCONF protocol conformance are supported for contents of the <rpc> elements:

- All XML attributes in the <rpc> start tag will be returned to the client (unchanged) in the <rpc-reply> element. The order of the XML attributes may not be preserved.
- All XML namespace prefix assignments declared in the <rpc> element (via the 'xmlns' attribute) will be used within the <rpc-reply>, and most descendant nodes of the <rpc-reply. The exception is the <error-path> element, which may use the default XML prefix for a given module, by declaring a new xmlns attribute for the namespace.
- The so-called mandatory 'message-id' attribute is ignored by the server, along will all other XML attributes in the <rpc> element. The server will not generate an error if this attribute is missing, as specified in RFC 4741. The new version of the NETCONF protocol removes this rule.

All <rpc> element contents must be declared within the proper namespace, except the contents of a subtree <filter> element for a <get> or <get-config> operation.

Access control will be enforced as follows:

- All <rpc> operation requests except <close-session> will be checked in the access control model (yuma-nacm.yang). This operation can always be invoked by any user, to allow graceful session termination in all cases.
- If the user name for the session matches the **--superuser** configuration parameter, then the operation will always be allowed.
- If the access control 'no-rule' default for RPC execution is set to 'permit', and there is no access control rule found to match the current <rpc> request, the the operation will always be allowed. The default for this parameter is 'permit'.
- If a matching access control rule is found, execution access will be permitted or denied, based on the specific rule. (I.e., 'exec' privilege bit set or not).

- If the operation reads or writes any database data, then the access control model will be checked again, for each database node specified in the request.
  - If the operation is requesting read access, then any nodes for which read permission is not granted will simply be skipped in the result. No error or warning will be reported.
  - If the operation is requesting write access, then any nodes for which write permission is not granted will cause an 'access-denied' error.

The server does not generate inline <rpc-error> elements at this time, for any runtime exceptions that occur while retrieving data for a <get>, <get-config>, or <copy-config> operation. Instead, unavailable nodes are just skipped.

A future version will support this feature, so managers should expect that <rpc-error> might appear within the data in a reply, not just a child node of the <rpc-reply> element.

2.4.6 Session Termination

A session can terminate for several reasons:

- <close-session> operation invoked
- <kill-session> operation invoked
- SSH session terminated unexpectedly
- TCP connection terminated unexpectedly

When a session terminates, the server does the following:

- will release any locks the session had (if any)
- will discard all changes in the <candidate> configuration, if this database was locked by the session.
- will remove the <session> list entry from the **/netconf-state/sessions** container
- will generate a <sysSessionEnd> notification entry for the closed or killed session

2.5 Error Reporting

All errors are reported using the standard <rpc-error> element.

If the operation does not return any data, then the <rpc-reply> element will either contain 1 <ok/> element, or 1 or more <rpc-error> elements.

If the operation returns any data (i.e., the YANG rpc definition for the operation has an 'output' section), then the <rpc-reply> element may have both <rpc-error> and data elements within it. If there were errors in the input, then only 1 or more <rpc-error> elements will be returned. It is possible that the required data will be returned, after any errors, but not likely.

The internal **netconfd** error code for each <rpc-error> is returned in an <error-info> extension called <error-number>.

Normally, the same <error-app-tag> and <error-message> values are returned for a specific error number. However, some YANG errors allow these fields to be user-defined. If there is a user-defined <error-app-tag> and/or <error-message> values, then they will be used instead of the default values.

This section describes the **netconfd** implementation details which may affect <rpc-error> processing by a client application.

2.5.1 <error-severity> Element

The <error-severity> field will always be set to 'error'. There are no warnings generated by **netconfd**.

2.5.2 <error-tag> Element

All NETCONF <error-tag> enumerations are supported, except 'partial-operation'. This error is being deprecated in the standard because nobody has implemented it.

If this field is set to 'invalid-value', then the <bad-value> element should be present in the <error-info>, identifying the invalid value that caused the problem.

All standard <error-info> contents are supported. The following table summarizes the different <error-tag> values. The <error-number> parameter is not shown in the 'error-info' column because it is added for every error-tag.

<error-tag> Summary

error-tag	error-info	description
access-denied	none	NACM denied access
bad-attribute	<bad-attribute> <bad-element> <bad-value>	just for the few attributes used in NETCONF
bad-element	<bad-element> <bad-value>	sometimes used instead of invalid-value
data-exists	none	nc:operation='create'
data-missing	none	nc:operation='delete' or 'replace'
in-use	none	edit on locked database
invalid-value	<bad-value>	for typedef constraints
lock-denied	<session-id>	for <lock> only
missing-attribute	<bad-attribute>	just for the few attributes used in NETCONF
missing-element	<bad-element>	mandatory parameters
operation-not-supported	none	unsupported, false if-feature inside rpc
operation-failed	none	when no other error-tag applies
partial-operation	<ok-element> <err-element> <noop-element>	not implemented
resource-denied	none	malloc failed
rollback-failed	none	rollback-on-error failed
too-big	none	input too big to buffer
unknown-attribute	<bad-attribute> <bad-element>	for any non-NETCONF attributes found
unknown-element	<bad-element>	wrong element name in a known namespace

unknown-namespace	<bad-element>	module not loaded

2.5.3 <error-app-tag> Element

The <error-app-tag> field provided a more specific error classification than the <error-tag> field. It is included in every <rpc-error> response.

This field is encoded as a simple string.

It is possible that error-app-tag values from different vendors will use the same string. A client application needs to account for this shared namespace.

If the YANG 'error-app-tag' statement is defined for the specific error that occured, then it will be used instead of the default value.

The following table describes the default <error-app-tag> values used by **netconfd**:

<error-app-tag> Summary

error-app-tag	description
data-incomplete	the input parameters are incomplete
data-invalid	one or more input parameters are invalid
data-not-unique	unique statement violation (YANG, sec. 13.1)
duplicate-error	trying to create a duplicate list or leaf-list entry
general-error	no other description fits
instance-required	missing mandatory node (YANG, sec. 13.5)
internal-error	internal software error
io-error	NETCONF session IO error
libxml2-error	libxml2 internal error or parsing error
limit-reached	some sort of defined limit was reached
malloc-error	malloc function call failed
missing-choice	mandatory choice not found (YANG, sec. 13.6)
missing-instance	mandatory leaf not found (YANG, sec. 13.7)
must-violation	must expression is false (YANG, sec. 13.4)
no-access	access control violation
no-matches	<get-schema> module or revision search failed
no-support	operation or sub-operation not implemented
not-in-range	YANG range test failed
not-in-value-set	YANG enumeration or bits name is invalid
pattern-test-failed	YANG pattern test failed
recover-failed	commit or rollback-on-error failed to leave the target database unchanged
resource-in-use	in-use error or <create-subscription> while a subscription is already active

ssh-error	SSH transport error
too-few-elements	min-elements violation (YANG, sec. 13.3)
too-many-elements	max-elements violation (YANG, sec. 13.2)

2.5.4 <error-path> Element

The <error-path> field indicates the node that caused the error.

It is encoded as a YANG instance-identifier.

If the node that caused the error is within the incoming <rpc> request, then the error path will start with the <rpc> element, and contain all the node identifiers from this document root to the node that caused the error.

```
<error-path>
    /nc:rpc/nc:edit-
config/nc:config/nacm:nacm/nacm:groups/nacm:group
</error-path>
```

The 'xmlns' attributes which define the 'nc' and 'nacm' prefixes would be present in the <rpc-reply> or the <rpc-error> element start tags.

If the node that caused the error is **not** within the incoming <rpc> request, then the error path will start with the top-level YANG module element that contains the error, not the <rpc> element.

This extended usage of the <error-path> field is defined in the YANG specification, not the NETCONF specification. This situation will occur if <validate> or <commit> operations detect errors. It can also occur if the <test-option> for the <edit-config> operation is 'test-then-set', and errors unrelated to the provided in-line <config> content are reported. and contain all the node identifiers from this document root to the node that caused the error.

```
<error-path>
    /nacm:nacm/nacm:groups/nacm:group[name='admin']/groupIdentity
</error-path>
```

2.5.5 <error-message> Element

The <error-message> field provides a short English language description of the error that usually corresponds to the <error-number> field.

If the YANG <error-message> statement is available for the error that occurred, it will be used instead of the default error message.

2.5.6 <error-info> Element

The <error-info> container is used to add error-specific data to the error report.

There are some standard elements that are returned for specific errors, and some elements specific to the **netconfd** server.

<error-info> Summary

child node	description
<bad-attribute>	name of the XML attribute that caused the error
<bad-element>	name of the element that caused the error or contains the attribute that caused the error
<bad-value>	value that caused the error
<error-number>	internal error number for the error condition
<missing-choice>	name of the missing mandatory YANG choice
<session-id>	session number of the current lock holder

2.5.7 INSTANCE-REQUIRED ERROR EXAMPLE

The instance-required error-app-tag is generated when a YANG leaf is mandatory, but was not set. An error will be returned right away if the target is the <running> configuration, or if the (default) **test-then-set** option is used for the <test-option>. Otherwise, this error is generated when the <commit> operation is invoked.

instance-required

data	description
error-tag	data-missing
error-app-tag	instance-required
error-path	identifies the leaf that is missing
error-info	none
error-number	310

Example Request:

- create /test2 (?s used to skip the mandatory <a2> leaf, but the <foo> leaf is set)

```
<?xml version="1.0" encoding="UTF-8"?>
<nc:rpc xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0"
   message-id="2">
   <nc:edit-config>
      <nc:target>
         <nc:candidate/>
      </nc:target>
      <nc:default-operation>none</nc:default-operation>
      <nc:config>
         <t:test2  nc:operation="create"
            xmlns:t="http://netconfcentral.org/ns/test">
            <t:foo>xxx</t:foo>
         </t:test2>
      </nc:config>
   </nc:edit-config>
</nc:rpc>
```

Example Error Reply:

```
<?xml version="1.0" encoding="UTF-8"?>
<nc:rpc-reply xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0"
   message-id="2" xmlns:t="http://netconfcentral.org/ns/test"
   xmlns:ncx="http://netconfcentral.org/ncx">
   <nc:rpc-error>
      <nc:error-type>application</nc:error-type>
      <nc:error-tag>data-missing</nc:error-tag>
      <nc:error-severity>error</nc:error-severity>
      <nc:error-app-tag>instance-required</nc:error-app-tag>
      <nc:error-path>/t:test2/t:a2</nc:error-path>
      <nc:error-message xml:lang="en">required value instance not
found</nc:error-message>
      <nc:error-info>
         <ncx:error-number>310</ncx:error-number>
      </nc:error-info>
   </nc:rpc-error>
</nc:rpc-reply>
```

2.5.8 MISSING-CHOICE ERROR EXAMPLE

The missing-choice error is generated when a YANG choice is mandatory, but no case from the choice was set. An error will be returned right away if the target is the <running> configuration, or if the (default) **test-then-set** option is used for the <test-option>. Otherwise, this error is generated when the <commit> operation is invoked.

missing-choice error

data	description
error-tag	data-missing
error-app-tag	missing-choice
error-path	identifies the parent of the missing choice
error-info	<missing-choice>
error-number	296

Example Request:

- create --target=/musttest (?s skip the mandatory choice)

```
<?xml version="1.0" encoding="UTF-8"?>
<nc:rpc xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0"
   message-id="2">
   <nc:edit-config>
      <nc:target>
         <nc:candidate/>
      </nc:target>
      <nc:default-operation>none</nc:default-operation>
      <nc:config>
         <t:musttest  nc:operation="create"
            xmlns:t="http://netconfcentral.org/ns/test"/>
      </nc:config>
   </nc:edit-config>
</nc:rpc>]
```

Example Error Reply:

```
<?xml version="1.0" encoding="UTF-8"?>
<nc:rpc-reply xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0"
   message-id="2" xmlns:t="http://netconfcentral.org/ns/test"
   xmlns:ncx="http://netconfcentral.org/ncx">
   <nc:rpc-error xmlns:y="urn:ietf:params:xml:ns:yang:1">
      <nc:error-type>application</nc:error-type>
      <nc:error-tag>data-missing</nc:error-tag>
      <nc:error-severity>error</nc:error-severity>
      <nc:error-app-tag>missing-choice</nc:error-app-tag>
      <nc:error-path>/t:musttest</nc:error-path>
      <nc:error-message xml:lang="en">missing mandatory
choice</nc:error-message>
      <nc:error-info>
         <y:missing-choice
xmlns:y="urn:ietf:params:xml:ns:yang:1">musttest</y:missing-choice>
         <ncx:error-number>296</ncx:error-number>
      </nc:error-info>
   </nc:rpc-error>
</nc:rpc-reply>
```

2.5.9 NO-MATCHES ERROR EXAMPLE

The no-matches error is generated when parameters for the <get-schema> operation identify a non-existent YANG file.

No Matches

data	description
error-tag	operation-failed
error-app-tag	no-matches
error-path	identifies the <identifier> or <revision> parameter in the <get-schema> request
error-info	<bad-value>
error-number	365

Example Request:

- get-schema identifier=foo version= format=yang

```
<?xml version="1.0" encoding="UTF-8"?>
<nc:rpc xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0"
   message-id="3">
   <ns:get-schema xmlns:ns="urn:ietf:params:xml:ns:netconf:state">
      <ns:identifier>foo</ns:identifier>
      <ns:version/>
      <ns:format>ns:yang</ns:format>
   </ns:get-schema>
</nc:rpc>
```

Example Error Reply:

```
<?xml version="1.0" encoding="UTF-8"?>
<nc:rpc-reply xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0"
   message-id="3" xmlns:ns="urn:ietf:params:xml:ns:netconf:state"
   xmlns:ncx="http://netconfcentral.org/ncx">
   <nc:rpc-error>
      <nc:error-type>protocol</nc:error-type>
      <nc:error-tag>operation-failed</nc:error-tag>
      <nc:error-severity>error</nc:error-severity>
      <nc:error-app-tag>no-matches</nc:error-app-tag>
      <nc:error-path>/nc:rpc/ns:get-
schema/ns:input/ns:identifier</nc:error-path>
      <nc:error-message xml:lang="en">no matches found</nc:error-
message>
      <nc:error-info>
         <ncx:bad-value>foo</ncx:bad-value>
         <ncx:error-number>365</ncx:error-number>
      </nc:error-info>
   </nc:rpc-error>
</nc:rpc-reply>
```

2.5.10 NOT-IN-RANGE ERROR EXAMPLE

The not-in-range error is generated when a numeric type viaolates a YANG range statement.

not-in-range

data	description
error-tag	invalid-value
error-app-tag	not-in-range
error-path	identifies the leaf or leaf-list node that is not in range
error-info	<bad-value>
error-number	288

Example Request:

- create /int8.1 value=1000

```
<?xml version="1.0" encoding="UTF-8"?>
<nc:rpc xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0"
   message-id="4">
   <nc:edit-config>
      <nc:target>
         <nc:candidate/>
      </nc:target>
      <nc:default-operation>none</nc:default-operation>
      <nc:config>
         <t:int8.1  nc:operation="create"
            xmlns:t="http://netconfcentral.org/ns/test">1000</t:int8
.1>
      </nc:config>
   </nc:edit-config>
</nc:rpc>
```

Example Error Reply:

```
<?xml version="1.0" encoding="UTF-8"?>
<nc:rpc xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0"
   message-id="4">
   <nc:edit-config>
      <nc:target>
         <nc:candidate/>
      </nc:target>
      <nc:default-operation>none</nc:default-operation>
      <nc:config>
         <t:int8.1  nc:operation="create"
            xmlns:t="http://netconfcentral.org/ns/test">1000</t:int8
.1>
      </nc:config>
```

```
   </nc:edit-config>
</nc:rpc>
```

2.6 Protocol Operations

This section describes the **netconfd** implementation details that may affect usage of the NETCONF protocol operations.

Every protocol operation is defined with a YANG rpc statement.

All NETCONF operations and several proprietary operations are supported,

2.6.1 <close-session>

The <close-session> operation is always allowed, even if access control rules exist which somehow disallow 'exec' privileges to a session for this operation.

A <sysSessionEnd> notification with a <terminationReason> field set to 'closed' will be generated when this operation is invoked.

<close-session> operation

Min parameters:	0
Max parameters:	0
Return type:	status
YANG file:	yuma-netconf.yang
Capabilities needed:	none

Mandatory Parameters:

- none

Optional Parameters:

- none

Returns:

- <ok/>; an <rpc-reply> will be sent to the session before terminating the session.

Possible Operation Errors:

- none

Example Request:

```
<?xml version="1.0" encoding="UTF-8"?>
<nc:rpc xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0"
   message-id="2">
   <nc:close-session/>
</nc:rpc>
```

Example Reply:

```
<?xml version="1.0" encoding="UTF-8"?>
<nc:rpc-reply xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0"
   message-id="2">
   <nc:ok/>
</nc:rpc-reply>
```

2.6.2 <COMMIT>

The <commit> operation is only available when the **:candidate** capability is supported.

This operation causes all the edits in the <candidate> configuration to be applied to the <running> configuration. If there are no edits, then this operation has no affect.

If multiple sessions have made edits to the <candidate> configuration (because locking was not used), then all these edits will be applied at once, not just the edits from the current session.

It the <candidate> or <running> configurations are locked by another session, then this operation will fail with an 'in-use' error.

Normally, if there have been no changes made to the <candidate> configuration, then this operation has no effect. An <ok/> response will be returned without altering the <running> configuration.

However, if the <running> configuration encountered any errors during the initial load from NV-storage (such as startup-cfg.xml), then the current contents of the <running> configuration will be written to NV-storage, even if there are no changes to the <candidate> configuration.

The :confirmed-commit capability is not supported yet, so the <confirmed> and <timeout> parameters are not available at this time.

<commit> operation

Min parameters:	0
Max parameters:	0
Return type:	status
YANG file:	yuma-netconf.yang
Capabilities needed:	:candidate

Mandatory Parameters:

- none

Optional Parameters:

- none

Returns:

- <ok/>

Possible Operation Errors:

- access-denied: access control configured to deny access to this operation
- in-use: the <candidate> or <running> configuration is locked by another session.

- operation-failed
  - must-violation: must statement is false
  - too-few-elements: min-elements expression is false
  - too-many-elements: max-elements expression is false
  - data-not-unique: unique statement violation
- data-missing: mandatory statement violation or missing leafref path object

Example Request:

```
<?xml version="1.0" encoding="UTF-8"?>
<nc:rpc xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0"
   message-id="5">
   <nc:commit/>
</nc:rpc>
```

Example Reply:

```
<?xml version="1.0" encoding="UTF-8"?>
<nc:rpc-reply xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0"
   message-id="5">
   <nc:ok/>
</nc:rpc-reply>
```

2.6.3 <COPY-CONFIG>

The <copy-config> operation is used to transfer entire configuration databases in one operation.

This is a destructive 'stop-on-error' operation. It is not like <edit-config> or <commit>, which can be used in an 'all-or-nothing' manner.

A failed <copy-config> can leave the target of the operation in an unstable, invalid state. This operation should be used with caution.

The <source> and <target> parameters are simple to understand, but there are many interactions and some complexity, due to so many combinations of optional capabilities that are possible.

When in-line configuration data is used in the <source> parameter, it is applied to the <target> differently, depending on the database.

- If the <target> is the <startup> configuration, then the new configuration simply overwrites the old one, and no validation is done at all.
- If the <target> is the <candidate> or <running> configuration, then the new configuration is applied as if the operation was an <edit-config> operation with a <default-operation> parameter set to 'replace'. All validation and access control procedures are followed.

The <with-defaults> parameter is also available for filtering the output as it is copied to the target.

<copy-config> operation

Min parameters:	2
Max parameters:	3
Return type:	status
YANG file:	yuma-netconf.yang
Capabilities needed:	none
Capabilities optional:	:candidate :writable-running :startup

Mandatory Parameters:

- **source**
  - type: container with 1 of N choice of leafs
  - usage: mandatory
  - default: none
  - This parameter specifies the name of the source database for the copy operation.
  - container contents: 1 of N:
    - **candidate**
      - type: empty
      - capabilities needed: :candidate
    - **running**
      - type: empty
      - capabilities needed: none
    - **startup**
      - type: empty
      - capabilities needed: startup
    - **config:**
      - type: container (in-line configuration data)
      - capabilities needed: none
        - The **netconfd** server will only the superuser account is allowed to copy in-line data into a database. All other users will get an 'access'-denied' error. Access to the entire database is required for this operational mode.
- **target**
  - type: container with 1 of N choice of leafs
  - usage: mandatory
  - default: none
  - This parameter specifies the name of the target database for the copy operation.
  - container contents: 1 of N:
    - **candidate**

        - type: empty
        - capabilities needed: :candidate
    - **startup**
        - type: empty
        - capabilities needed: startup

Optional Parameters:

- **with-defaults**
    - type: enumeration (none report-all trim explicit)
    - default: none
    - capabilities needed: with-defaults
    - This parameter controls how nodes containing only default values are copied to the target database. If the <target> parameter is <candidate>, then this parameter will be ignored.

Returns:

- <ok/>

Possible Operation Errors:

- access-denied**:** access control configured to deny access to this operation
- in-use: the configuration indicated by the <target> parameter is locked by another session.
- <commit> errors, if the **target** parameter is the <running> configuration

Example Request:

```
<?xml version="1.0" encoding="UTF-8"?>
<nc:rpc xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0"
   message-id="7">
   <nc:copy-config>
      <nc:source>
         <nc:running/>
      </nc:source>
      <nc:target>
         <nc:startup/>
      </nc:target>
   </nc:copy-config>
</nc:rpc>
```

Example Reply:

```
<?xml version="1.0" encoding="UTF-8"?>
<nc:rpc-reply xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0"
   message-id="7">
   <nc:ok/>
</nc:rpc-reply>
```

2.6.4 <CREATE-SUBSCRIPTION>

The <create-subscription> operation is used to start a NETCONF notifications subscription.

Only the 'NETCONF' stream is supported by **netconfd**.

A replay subscription is created by including the <startTime> parameter.

The subscription will continue until the session is closed, unless the <stopTime> parameter is present. In that case, the subscription will terminate at that tim (if in the future) or when all replay notifications with a lower <eventTime> value have been delivered.

The :notification and :interleave capabilities are always supported by netconfd.

The replay notification feature can be controlled with the --eventlog-size configuration parameter. If this is set to '0', then no stored notifications will be available for replay. The default is store the most recent 1000 system notification events.

An entry will be created in the 'subscriptions' data structure in the ietf-netconf-monitoring module, when the subscription is successfully started.

<create-subscription> operation

Min parameters:	0
Max parameters:	4
Return type:	status
YANG file:	notifications.yang
Capabilities needed:	:notification
Capabilities optional:	:interleave

Mandatory Parameters:

- none

Optional Parameters:

- **stream**
  - type: string
  - default: 'NETCONF'
  - This parameter specifies the name of the notification stream for this subscription request. Only the 'NETCONF' stream is supported.
- **filter**
  - type: anyxml (same as the <get> or <get-config> filter parameter)
  - default: none
  - This parameter specifies a boolean filter that should be applied to the stream. This is the same format as the standard <filter> element in RFC 4741, except that instead of creating a subset of the database for an <rpc-reply> PDU, the filter is used as a boolean test to send or drop each notification delivered from the server.
    - If any nodes are left in the 'test response', the server will send the entire notification.
    - If the result is empty after the filter is applied to the “test response”, then the server will not send the notification at all.
    - It is possible that access control will either cause the a notification to be dropped entirely, or may be pruned and still delivered. The standard is not clear on this topic.

The **netconfd** server will prune any unauthorized payload from an eventType, but if the <eventType> itself is unauthorized, the entire notification will be dropped.

- **startTime**
  - type: yang:date-and-time
  - This parameter causes any matching replay notifications to be delivered by the server, if notification replay is supported by the server. A notification will match if its <eventTime> value is greater or equal to the value of this parameter.
- **stopTime**
  - type: yang:date-and-time
  - usage: not allowed unless startTime is also present
  - This parameter causes any matching replay notifications to be delivered by the server, if notification replay is supported by the server. A notification will match if its <eventTime> value is less than the value of this parameter.
    - This parameter must be greater than the **startTime** parameter, or an error will be returned by the server.

Returns:

- <ok/>

Possible Operation Errors:

- access-denied
- subscription already active

Example Request:

```
<?xml version="1.0" encoding="UTF-8"?>
<nc:rpc xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0"
   message-id="2">
   <ncEvent:create-subscription
      xmlns:ncEvent="urn:ietf:params:xml:ns:netconf:notification:1.0
">
      <ncEvent:startTime>2009-01-01T00:00:00Z</ncEvent:startTime>
   </ncEvent:create-subscription>
</nc:rpc>
```

Example Reply:

```
<?xml version="1.0" encoding="UTF-8"?>
<nc:rpc-reply xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0"
   message-id="2">
   <nc:ok/>
</nc:rpc-reply>
```

2.6.5 <DELETE-CONFIG>

The <delete-config> operation is used to delete the entire contents of a NETCONF database.

By default, only the superuser account is allowed to invoke this operation.

If the **:startup** capability is supported, then the <startup> configuration can be cleared. This will affect the startup file that was actually loaded into the server, or the default file if the **--no-startup** configuration parameter was used.

The <running> and <candidate> configurations cannot be deleted.

The <startup> configuration can be repopulated with the <copy-config> or <commit> operations.

<delete-config> operation

Min parameters:	1
Max parameters:	1
Return type:	status
YANG file:	yuma-netconf.yang
Capabilities needed:	:startup

Mandatory Parameters:

- **target**
  - type: container with 1 of N choice of leafs
  - This parameter specifies the name of the target database for the <delete-config> operation.
  - container contents: 1 empty leaf value supported:
    - **startup**

Optional Parameters:

- none

Returns:

- <ok/>

Possible Operation Errors:

- access-denied
- in-use: <startup> database is locked

Example Request:

```
<?xml version="1.0" encoding="UTF-8"?>
<nc:rpc xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0"
   message-id="2">
   <nc:delete-config>
      <nc:target>
         <nc:startup/>
      </nc:target>
   </nc:delete-config>
</nc:rpc>
```

Example Reply:

```
<?xml version="1.0" encoding="UTF-8"?>
<nc:rpc-reply xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0"
   message-id="2">
   <nc:ok/>
</nc:rpc-reply>
```

2.6.6 <DISCARD-CHANGES>

The <discard-changes> operation is used to remove any edits from the <candidate> configuration. This is done by deleting the contents of the <candidate> and re-filling it with the contents of the <running> configuration.

If the <candidate> configuration is locked by another session, this operation will fail.

<discard-changes> operation

Min parameters:	0
Max parameters:	0
Return type:	status
YANG file:	yuma-netconf.yang
Capabilities needed:	:candidate

Mandatory Parameters:

- none

Optional Parameters:

- none

Returns:

- <ok/>

Possible Operation Errors:

- access-denied

Example Request:

```
<?xml version="1.0" encoding="UTF-8"?>
<nc:rpc xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0"
   message-id="3">
   <nc:discard-changes/>
</nc:rpc>
```

Example Reply:

```
<nc:rpc-reply xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0"
   message-id="3">
   <nc:ok/>
</nc:rpc-reply>
```

2.6.7 <EDIT-CONFIG>

The <edit-config> operation is used to alter the target database.

The target database is the <candidate> configuration if the **--target** configuration parameter is set to 'candidate', and the <running> configuration if it is set to 'running'.

The **nc:operation** attribute must appear within the inline <config> parameter contents if the <default-operation> parameter is set to 'none', or the <edit-config> operation will have no affect. This is not an error condition.

If the **nc:operation** attribute is used, then it may appear any number of times, and be arbitrarily nested within the <config> parameter contents. Certain combinations will cause errors, however, so this must be done carefully. For example, a 'delete' operation nested within a 'create' operation is an error, because the conditions for both operations cannot possibly be satisfied at once. Other combinations, such as 'merge' within 'create' are not an error because there are no conflicting conditions present for either operation.

<edit-config> operation

Min parameters:	2
Max parameters:	5
Return type:	status
YANG file:	yuma-netconf.yang
Capabilities needed:	:candidate or :writable-running
Capabilities optional:	:rollback-on-error :validate

Mandatory Parameters:

- **config**
  - type: anyxml
  - This parameter specifies the subset of the database that should be changed.
- **target**
  - type: container with 1 of N choice of leafs
  - This parameter specifies the name of the target database for the edit operation.
  - container contents: choice: 1 of N:
    - **candidate**
      - type: empty
      - capabilities needed: :candidate
    - **running**

- type: empty
- capabilities needed: :writable-running

Optional Parameters:

- **default-operation**
  - type: enumeration (merge replace none)
  - default: merge
  - This parameter specifies which edit operation will be in effect at the start of the operation, before any **nc:operation** attribute is found.
- **error-option**
  - type: enumeration (stop-on-error continue-on-error rollback-on-error
  - default: stop-on-error
  - This parameter specifies what the server should do when an error is encountered.
-
- **test-option**
  - type: enumeration (test-then-set set test-only)
  - default: 'test-then-se't if **:validate** capability is supported and 'set' otherwise

Returns:

- <ok/>

Possible Operation Errors:

- access-denied
- in-use (target database is locked by another session)
- <commit> validation errors: if **test-then-set** is used or the target is the <running> configuration.

Example Request:

```
<?xml version="1.0" encoding="UTF-8"?>
<nc:rpc xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0"
   message-id="12">
   <nc:edit-config>
      <nc:target>
         <nc:candidate/>
      </nc:target>
      <nc:default-operation>merge</nc:default-operation>
      <nc:test-option>set</nc:test-option>
      <nc:error-option>rollback-on-error</nc:error-option>
      <nc:config>
         <t:musttest xmlns:t="http://netconfcentral.org/ns/test">
            <t:A  nc:operation="create">'testing one two'</t:A>
         </t:musttest>
      </nc:config>
   </nc:edit-config>
</nc:rpc>
```

Example Reply:

```
<?xml version="1.0" encoding="UTF-8"?>
<nc:rpc-reply xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0"
   message-id="12">
   <nc:ok/>
</nc:rpc-reply>
```

2.6.8 <GET>

The <get> operation is used to retrieve data from the <running> configuration, or non-configuration data available on the server.

The simple command (<get/>) will cause all available data to be retrieved from the server. This may be generate a large response and waste resources.

To select only specific subsets of all available data, use subtree or XPath filtering by providing a <filter> parameter. Namespace prefixes are optional to use in XPath expressions. The netconfd will figure out the proper namespace, if possible. If prefixes are used, then they must be valid XML prefixes with a namespace properly declared in the PDU.

The retrieval of leaf or leaf-list nodes with default values is controlled with the <with-defaults> parameter.

If a portion of the requested data is not available due to access control restrictions, then that data is silently dropped from the <rpc-reply> message. It is implicitly understood that the client is only requesting data for which it is authorized to receive, in the event such data is selected in the request.

<get> operation

Min parameters:	0
Max parameters:	2
Return type:	data
YANG file:	yuma-netconf.yang
Capabilities needed:	none
Capabilities optional:	:candidate :startup :with-defaults

Mandatory Parameters:

- none

Optional Parameters:

- **filter**
  - **type='subtree'**
    - type: anyxml
    - This parameter specifies the subset of the database that should be retrieved.
  - **type='xpath' select='expr'**
    - type: empty
    - The unqualified **select** attribute is used to specify an XPath filter.

- **with-defaults**
  - type: enumeration (none report-all trim explicit)
  - usage: **--default-style** configuration parameter used as the default if no value is provided
  - This parameter controls how default leaf and leaf-list nodes are returned by the server.

Returns:

- <data> element

Possible Operation Errors:

- access-denied

Example subtree Filter Request:

```
<?xml version="1.0" encoding="UTF-8"?>
<nc:rpc xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0"
   message-id="4">
   <nc:get>
      <nc:filter type="subtree">
         <proc:proc xmlns:proc="http://netconfcentral.org/ns/proc">
            <proc:cpuinfo>
               <proc:cpu>
                  <proc:cpu_MHz/>
               </proc:cpu>
            </proc:cpuinfo>
         </proc:proc>
      </nc:filter>
   </nc:get>
</nc:rpc>
```

Equivalent XPath Filter Request:

```
<?xml version="1.0" encoding="UTF-8"?>
<nc:rpc xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0"
   message-id="4">
   <nc:get>
      <nc:filter type="xpath" select="/proc/cpuinfo/cpu/cpu_MHz"/>
   </nc:get>
</nc:rpc>]
```

Example Reply:

```
<?xml version="1.0" encoding="UTF-8"?>
<nc:rpc-reply xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0"
   message-id="4">
   <nc:data>
      <proc:proc xmlns:proc="http://netconfcentral.org/ns/proc">
         <proc:cpuinfo>
            <proc:cpu>
               <proc:cpu_MHz>1600.000</proc:cpu_MHz>
               <proc:processor>0</proc:processor>
            </proc:cpu>
            <proc:cpu>
               <proc:cpu_MHz>2667.000</proc:cpu_MHz>
               <proc:processor>1</proc:processor>
            </proc:cpu>
         </proc:cpuinfo>
      </proc:proc>
   </nc:data>
</nc:rpc-reply>
```

2.6.9 <GET-CONFIG>

The <get-config> operation is used to retrieve data from a NETCONF configuration database.

To select only specific subsets of all available data, use subtree or XPath filtering by providing a <filter> parameter. Namespace prefixes are optional to use in XPath expressions. The netconfd will figure out the proper namespace, if possible. If prefixes are used, then they must be valid XML prefixes with a namespace properly declared in the PDU.

The retrieval of leaf or leaf-list nodes with default values is controlled with the <with-defaults> parameter.

If a portion of the requested data is not available due to access control restrictions, then that data is silently dropped from the <rpc-reply> message. It is implicitly understood that the client is only requesting data for which it is authorized to receive, in the event such data is selected in the request.

<get-config> operation

Min parameters:	1
Max parameters:	3
Return type:	data
YANG file:	yuma-netconf.yang
Capabilities needed:	none
Capabilities optional:	:candidate :startup :with-defaults

Mandatory Parameters:

- **source**
  - type: container with 1 of N choice of leafs
  - usage: mandatory
  - This parameter specifies the name of the source database for the retrieval operation.
  - container contents: 1 of N:
    - **candidate**
      - type: empty
      - capabilities needed: :candidate
    - **running**
      - type: empty
      - capabilities needed: none
    - **startup**
      - type: empty
      - capabilities needed: startup

Optional Parameters:

- **filter**
  - **type='subtree'**

        - type: anyxml
        - This parameter specifies the subset of the database that should be retrieved.
    - **type='xpath' select='expr'**
        - type: empty
        - The unqualified **select** attribute is used to specify an XPath filter.
- **with-defaults**
    - type: enumeration (none report-all trim explicit)
    - usage: **--default-style** configuration parameter used as the default if no value is provided
    - This parameter controls how default leaf and leaf-list nodes are returned by the server.

Returns:

- <data> element

Possible Operation Errors:

- access-denied

Example subtree Filter Request:

```
<?xml version="1.0" encoding="UTF-8"?>
<nc:rpc xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0"
   message-id="3">
   <nc:get-config>
      <nc:source>
         <nc:candidate/>
      </nc:source>
      <nc:filter type="subtree">
         <nacm:nacm xmlns:nacm="http://netconfcentral.org/ns/nacm">
            <nacm:rules>
               <nacm:moduleRule/>
            </nacm:rules>
         </nacm:nacm>
      </nc:filter>
   </nc:get-config>
</nc:rpc>
```

Equivalent XPath Filter Request:

```
<?xml version="1.0" encoding="UTF-8"?>
<nc:rpc xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0"
   message-id="3">
   <nc:get-config>
      <nc:source>
         <nc:candidate/>
      </nc:source>
      <nc:filter type="xpath" select="/nacm/rules/moduleRule"/>
   </nc:get-config>
</nc:rpc>
```

Example Reply:

```
<?xml version="1.0" encoding="UTF-8"?>
<nc:rpc-reply xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0"
   message-id="3">
   <nc:data>
      <nacm:nacm xmlns:nacm="http://netconfcentral.org/ns/nacm">
         <nacm:rules>
            <nacm:moduleRule>
               <nacm:moduleName>netconf</nacm:moduleName>
               <nacm:allowed-rights>read exec</nacm:allowed-rights>
               <nacm:allowed-group>nacm:guest</nacm:allowed-group>
            </nacm:moduleRule>
            <nacm:moduleRule>
               <nacm:moduleName>netconfd</nacm:moduleName>
               <nacm:allowed-rights>read write exec</nacm:allowed-
rights>
               <nacm:allowed-group>nacm:admin</nacm:allowed-group>
               <nacm:comment>access to shutdown and restart
rpcs</nacm:comment>
            </nacm:moduleRule>
            <nacm:moduleRule>
               <nacm:moduleName>netconf</nacm:moduleName>
               <nacm:allowed-rights>read write exec</nacm:allowed-
rights>
               <nacm:allowed-group>nacm:admin</nacm:allowed-group>
               <nacm:allowed-group>nacm:monitor</nacm:allowed-group>
            </nacm:moduleRule>
         </nacm:rules>
      </nacm:nacm>
   </nc:data>
</nc:rpc-reply>
```

2.6.10 <GET-MY-SESSION>

The <get-my-session> operation is used to retrieve session customization data for the current session.

The session indent amount, line size, and default behavior for the with-defaults parameter can be controlled at this time.

<get-my-session> operation

Min parameters:	0
Max parameters:	0
Return type:	data
YANG file:	yuma-mysession.yang
Capabilities needed:	none

Mandatory Parameters:

- none

Optional Parameters:

- none

Returns:

- **<indent>**
  - type: uint32 (range 0 .. 9)
  - This parameter specifies the desired indent amount for the session.
- **<linesize>**
  - type: uint32 (range 40 .. 1024)
  - This parameter specifies the desired line length for the session.
- **<with-defaults>**
  - type: enumeration (report-all, trim, explicit)
  - This parameter specifies the desired default with-defaults behavior for the session. The server-wide default value is set with the **--default-style** configuration parameter.

Possible Operation Errors:

- access-denied

Example Request:

```
<?xml version="1.0" encoding="UTF-8"?>
<nc:rpc xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0"
   message-id="2">
   <myses:get-my-session
xmlns:myses="http://netconfcentral.org/ns/mysession"/>
</nc:rpc>
```

Example Reply:

```
<?xml version="1.0" encoding="UTF-8"?>
<nc:rpc-reply xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0"
   message-id="2">
   <myses:indent
xmlns:myses="http://netconfcentral.org/ns/mysession">
      3
   </myses:indent>
   <myses:linesize
xmlns:myses="http://netconfcentral.org/ns/mysession">
      72
   </myses:linesize>
   <myses:with-defaults
xmlns:myses="http://netconfcentral.org/ns/mysession">
      report-all
   </myses:with-defaults>
</nc:rpc-reply>
```

2.6.11 <GET-SCHEMA>

The <get-schema> operation is used to retrieve YANG modules and submodules from the server.

The 'YANG' format is supported for all YANG files loaded into the server.

If the <version> parameter is set to the empty string, then the server will return whichever version it supports. If multiple versions are supported, then the server will pick a canonical version, which may not be the most recent version.

The <identifier> parameter must contain the name of the YANG file without any path or file extension specification.

<get-schema> operation

Min parameters:	3
Max parameters:	3
Return type:	data
YANG file:	yuma-netconf.yang
Capabilities needed:	:schema-retrieval

Mandatory Parameters:

- **identifier**
  - type: string
  - This parameter specifies the name of the module to retrieve.
    - Do not use any path specification of file extension; just the module name is entered.
    - The name is case-sensitive, and must be specified exactly as defined.
- **version**
  - type: string
  - This parameter specifies the version of the module to retrieve.
    - This will be the most recent YANG revision date string defined in a module revision statement.
    - If any version is acceptable, or if the specific version is not known, then use the empty string.
- **format**
  - type: enumeration (XSD YANG RNG)
  - This parameter specifies the format of the module to be retrieved. Only the YANG format is supported.
    - YANG: draft-ietf-netmod-yang

Returns:

- <data> element (contents of the YANG file)

Possible Operation Errors:

- access-denied

Example Request:

```
<?xml version="1.0" encoding="UTF-8"?>
<nc:rpc xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0"
   message-id="55">
   <ns:get-schema xmlns:ns="urn:ietf:params:xml:ns:netconf:state">
```

```
      <ns:identifier>ietf-with-defaults</ns:identifier>
      <ns:format>YANG</ns:format>
      <ns:version/>
   </ns:get-schema>
</nc:rpc>
```

Example Reply:

```
<?xml version="1.0" encoding="UTF-8"?>
<nc:rpc-reply xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0"
   message-id="55">
   <ns:data xmlns:ns="urn:ietf:params:xml:ns:netconf:state">
*** entire contents of YANG module would be here, no extra indenting
***
   </ns:data>
</nc:rpc-reply>
```

2.6.12 <KILL-SESSION>

The <kill-session> operation is used to force the termination of another NETCONF session. This is sometimes needed if an idle session which is holding one or more locks was abandoned. It may also be needed for security reasons. In any case, this operation should be used with extreme caution.

<kill-session> operation

Min parameters:	1
Max parameters:	1
Return type:	status
YANG file:	yuma-netconf.yang
Capabilities needed:	none

Mandatory Parameters:

- **session-id**
  - type: uint32 (range: 1.. max)
  - default: none
  - This parameter specifies the session number of the currently active NETCONF session that should be terminated.

Optional Parameters:

- none

Returns:

- <ok/>

Possible Operation Errors:

- access-denied

Example Request:

```
<?xml version="1.0" encoding="UTF-8"?>
<nc:rpc xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0"
   message-id="42">
   <nc:kill-session>
      <nc:session-id>1</nc:session-id>
   </nc:kill-session>
</nc:rpc>
```

Example Reply:

```
<?xml version="1.0" encoding="UTF-8"?>
<nc:rpc-reply xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0"
   message-id="42">
   <nc:ok/>
</nc:rpc-reply>
```

2.6.13 <LOAD>

The <load> operation is used to load new YANG modules at run-time.

The module file must already be present in the module search path of the server.

There must not be any version of the module already loaded.

This operation is tagged as **nacm:secure**, so by default, only the super user account is allowed to use it.

<load> operation

Min parameters:	1
Max parameters:	3
Return type:	data
YANG file:	yuma-system.yang
Capabilities needed:	none

Mandatory Parameters:

- **module**
  - The name of the YANG module to load

Optional Parameters:

- **revision**
  - The revision date of the module to load. If missing, then server can select the version to use.

- **deviation**
  - The name of a deviation module to load before loading this module. Zero or more instances of this parameter can be entered. Duplicates will be ignored.

Returns:

- <mod-revision>
  - The revision date of the module that was already loaded, or was just loaded.

Possible Operation Errors:

- access-denied
- module not found
- version not found
- different version already loaded
- module parsing errors
- resource errors

Example Request:

```
<?xml version="1.0" encoding="UTF-8"?>
<nc:rpc xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0"
   message-id="52">
   <nd:load xmlns:nd="http://netconfcentral.org/ns/netconfd">
      <nd:module>test2</nd:module>
   </nd:load>
</nc:rpc>
```

Example Reply:

```
<?xml version="1.0" encoding="UTF-8"?>
<nc:rpc-reply xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0"
   message-id="52">
   <nd:mod-revision
xmlns:nd="http://netconfcentral.org/ns/netconfd">
       2008-10-15
   </nd:mod-revision>
</nc:rpc-reply>
```

2.6.14 <lock>

The <lock> operation is used to insure exclusive write access to an entire configuration database.

The <running> configuration can be locked at any time, if it is currently unlocked.

If the **:startup** capability is supported, the <startup> configuration can be locked at any time, if it is currently unlocked.

If the :candidate capability is supported, and it is not already locked, then it may usually be locked. However, the <candidate> configuration can only be locked if there are no edits already contained within it. A <discard-changes> operation may be needed to clear any leftover edits, if this operation fails with a 'resource-denied' error instead of a 'lock-denied' error.

If the session holding the lock is terminated for any reason, the lock will be released, as if the <unlock> operation was invoked.

<lock> operation

Min parameters:	1
Max parameters:	1
Return type:	status
YANG file:	yuma-netconf.yang
Capabilities needed:	none
Capabilities optional:	:candidate :startup

Mandatory Parameters:

- **target**
  - type: container with 1 of N choice of leafs
  - This parameter specifies the name of the target database to be locked.
  - container contents: 1 of N:
    - **candidate**
      - type: empty
      - capabilities needed: :candidate
    - **running**
      - type: empty
      - capabilities needed: none
    - **startup**
      - type: empty
      - capabilities needed: startup

Optional Parameters:

- none

Returns:

- <ok/>

Possible Operation Errors:

- access-denied
- lock-denied (lock already held by <session-id> in the <error-info>)
- resource-denied (candidate is dirty; <discard-changes> needed)

Example Request:

```
<?xml version="1.0" encoding="UTF-8"?>
<nc:rpc xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0"
   message-id="62">
   <nc:lock>
      <nc:target>
         <nc:candidate/>
      </nc:target>
   </nc:lock>
</nc:rpc>
```

Example Reply:

```
<?xml version="1.0" encoding="UTF-8"?>
<nc:rpc-reply xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0"
   message-id="62">
   <nc:ok/>
</nc:rpc-reply>
```

2.6.15 <NO-OP>

The <no-op> operation is used for debugging and performance testing purposes.

This operation does not do anything. It simply returns <ok/>, unless any parameters are provided.

<no-op> operation

Min parameters:	0
Max parameters:	0
Return type:	status
YANG file:	yuma-system.yang
Capabilities needed:	none

Mandatory Parameters:

- none

Optional Parameters:

- none

Returns:

- <ok/>

Possible Operation Errors:

- access-denied

Example Request:

```
<?xml version="1.0" encoding="UTF-8"?>
<nc:rpc xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0"
   message-id="63">
   <nd:no-op xmlns:nd="http://netconfcentral.org/ns/netconfd"/>
</nc:rpc>
```

Example Reply:

```
<?xml version="1.0" encoding="UTF-8"?>
<nc:rpc-reply xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0"
   message-id="63">
   <nc:ok/>
</nc:rpc-reply>
```

2.6.16 <PARTIAL-LOCK>

The <partial-lock> operation is used to lock part of the <running> database.

Refer to RFC 5717 or the **ietf-netconf-partial-lock.yang** module for details on this operation.

<partial-lock> operation

Min parameters:	1
Max parameters:	1
Return type:	data
YANG file:	ietf-netconf-partial-lock.yang
Capabilities needed:	:partial-lock

Mandatory Parameters:

- **<select>**
  - type: XPath 1.0 string
  - This parameter contains an XPath expression for a node-set result, identifying the database nodes to lock.
  - One of more instances of this parameter is required.
  - The server allows relaxed XPath syntax. If prefixes are used, then a proper namespace declaration must be present in the request. If prefixes are not used, then any available namespace that matches the local name will be used.

Optional Parameters:

- none

Returns:

- **<lock-id>**
  - type: uint32 (range 1 .. MAX_UINT)
  - This data identifies the lock ID to use when releasing the lock with the <partial-unlock> operation.
  - There will be one of these elements in the reply.
- **<locked-node>**
  - type: instance-identifier
  - This data identifies a locked node as a result of the request.
  - There will be one or more of these elements in the reply.

Possible Operation Errors:

- access denied, in-use, resource-denied

Example Request:

```
<?xml version="1.0" encoding="UTF-8"?>
<nc:rpc xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0"
   message-id="260">
   <pl:partial-lock
xmlns:pl="urn:ietf:params:xml:ns:netconf:partial-lock:1.0">
      <pl:select>//interface</pl:select>
   </pl:partial-lock>
</nc:rpc>
```

Example Reply:

```
<?xml version="1.0" encoding="UTF-8"?>
<nc:rpc-reply xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0"
   message-id="260" xmlns:if="http://netconfcentral.org/ns/yuma-
interfaces">
   <pl:lock-id xmlns:pl="urn:ietf:params:xml:ns:netconf:partial-
lock:1.0">1</pl:lock-id>
   <pl:locked-node xmlns:pl="urn:ietf:params:xml:ns:netconf:partial-
lock:1.0">/if:interfaces/if:interface[if:name='virbr0']</pl:locked-
node>
   <pl:locked-node xmlns:pl="urn:ietf:params:xml:ns:netconf:partial-
lock:1.0">/if:interfaces/if:interface[if:name='eth0']</pl:locked-
node>
   <pl:locked-node xmlns:pl="urn:ietf:params:xml:ns:netconf:partial-
lock:1.0">/if:interfaces/if:interface[if:name='lo']</pl:locked-node>
</nc:rpc-reply>
```

2.6.17 <PARTIAL-UNLOCK>

The <partial-unlock> operation is used to unlock part of the <running> database that was previously locked with the <partial-lock> operation. Only the session that called <partial-lock> can release the lock with this operation.

Refer to RFC 5717 or the **ietf-netconf-partial-lock.yang** module for details on this operation.

<partial-unlock> operation

Min parameters:	1
Max parameters:	1
Return type:	status
YANG file:	ietf-netconf-partial-lock.yang
Capabilities needed:	:partial-lock

Mandatory Parameters:

- **<lock-id>**
  - type: unit32 (1 .. MAX_UINT)
  - This parameter contains the lock ID of the partial lock to release.
  - One of more instances of this parameter is required.
  - The server allows relaxed XPath syntax. If prefixes are used, then a proper namespace declaration must be present in the request. If prefixes are not used, then any available namespace that matches the local name will be used.

Optional Parameters:

- none

Returns:

- <ok/>

Possible Operation Errors:

- access denied, invalid-value

Example Request:

```
<?xml version="1.0" encoding="UTF-8"?>
<nc:rpc xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0"
   message-id="263">
   <pl:partial-unlock
xmlns:pl="urn:ietf:params:xml:ns:netconf:partial-lock:1.0">
      <pl:lock-id>1</pl:lock-id>
   </pl:partial-unlock>
</nc:rpc>
```

Example Reply:

```
<?xml version="1.0" encoding="UTF-8"?>
<nc:rpc-reply xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0"
   message-id="263">
   <nc:ok/>
```

```
</nc:rpc-reply>
```

2.6.18 <RESTART>

The <restart> operation is used to restart the **netconfd** server.

By default, only the 'superuser' account is allowed to invoke this operation.

If permission is granted, then the current NETCONF session will dropped, during the server restart.

<restart> operation

Min parameters:	0
Max parameters:	0
Return type:	none
YANG file:	yuma-system.yang
Capabilities needed:	none

Mandatory Parameters:

- none

Optional Parameters:

- none

Returns:

- none; session will be dropped upon success

Possible Operation Errors:

- access denied

Example Request:

```
<?xml version="1.0" encoding="UTF-8"?>
<nc:rpc xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0"
   message-id="63">
   <nd:restart xmlns:nd="http://netconfcentral.org/ns/netconfd"/>
</nc:rpc>
```

Example Reply:

```
[no reply will be sent; session will be dropped instead.]
```

2.6.19 <SET-LOG-LEVEL>

The <set-log-level> operation is used to configure the server logging verbosity level.

Only the designated superuser user can invoke this operation by default.

<set-log-level> operation

Min parameters:	1
Max parameters:	1
Return type:	status
YANG file:	yuma-system.yang
Capabilities needed:	none

Mandatory Parameters:

- **<log-level>**
  - type: enumeration (off, error, warn, info, debug, debug2, debug3)
  - default: none
  - This parameter specifies the server logging verbosity level.

Optional Parameters:

- none

Returns:

- <ok/>

Possible Operation Errors:

- access-denied

Example Request:

```
<?xml version="1.0" encoding="UTF-8"?>
<nc:rpc xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0"
   message-id="3">
   <myses:set-my-session
xmlns:myses="http://netconfcentral.org/ns/mysession">
      <myses:indent>4</myses:indent>
      <myses:linesize>64</myses:linesize>
      <myses:with-defaults>trim</myses:with-defaults>
   </myses:set-my-session>
</nc:rpc>
```

Example Reply:

```
<?xml version="1.0" encoding="UTF-8"?>
<nc:rpc-reply xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0"
   message-id="3">
   <nc:ok/>
</nc:rpc-reply>
```

2.6.20 <set-my-session>

The <set-my-session> operation is used to configure the session customization data for the current session.

The session indent amount, line size, and default behavior for the with-defaults parameter can be controlled at this time.

<set-my-session> operation

Min parameters:	0
Max parameters:	3
Return type:	status
YANG file:	yuma-mysession.yang
Capabilities needed:	none

Mandatory Parameters:

- none

Optional Parameters:

- **<indent>**
  - type: uint32 (range 0 .. 9)
  - default: 3 (can be changed with the **--indent** configuration parameter)
  - This parameter specifies the desired indent amount for the session.
- **<linesize>**
  - type: uint32 (range 40 .. 1024)
  - default: 72
  - This parameter specifies the desired line length for the session.
- **<with-defaults>**
  - type: enumeration (report-all, trim, explicit)
  - default: report-all (can be changed with the **--default-style** configuration parameter)
  - This parameter specifies the desired default with-defaults behavior for the session. The server-wide default value is set with the **--default-style** configuration parameter.

Returns:

- <ok/>

Possible Operation Errors:

- access-denied

Example Request:

```
<?xml version="1.0" encoding="UTF-8"?>
<nc:rpc xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0"
```

```
   message-id="3">
   <myses:set-my-session
xmlns:myses="http://netconfcentral.org/ns/mysession">
      <myses:indent>4</myses:indent>
      <myses:linesize>64</myses:linesize>
      <myses:with-defaults>trim</myses:with-defaults>
   </myses:set-my-session>
</nc:rpc>
```

Example Reply:

```
<?xml version="1.0" encoding="UTF-8"?>
<nc:rpc-reply xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0"
   message-id="3">
   <nc:ok/>
</nc:rpc-reply>
```

2.6.21 <SHUTDOWN>

The <shutdown> operation is used to shut down the **netconfd** server.

By default, only the 'superuser' account is allowed to invoke this operation.

If permission is granted, then the current NETCONF session will dropped, during the server shutdown.

<shutdown> operation

Min parameters:	0
Max parameters:	0
Return type:	none
YANG file:	yuma-system.yang
Capabilities needed:	none

Mandatory Parameters:

- none

Optional Parameters:

- none

Returns:

- none; session will be dropped upon success

Possible Operation Errors:

- access denied

Example Request:

```
<?xml version="1.0" encoding="UTF-8"?>
<nc:rpc xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0"
   message-id="1">
   <nd:shutdown xmlns:nd="http://netconfcentral.org/ns/netconfd"/>
</nc:rpc>
```

Example Reply:

```
[no reply will be sent; session will be dropped instead.]
```

2.6.22 <UNLOCK>

The <unlock> operation is used to release a global lock held by the current session.

The specified configuration database must be locked, or a 'no-access' <error-app-tag> and an <error-message> of 'wrong-config-state' will be returned.

If the <candidate> configuration contains any edits that have not been committed, then these edits will all be lost if the <unlock> operation is invoked. A <discard-changes> operation is performed automatically by the server when the <candidate> database is unlocked.

<unlock> operation

Min parameters:	1
Max parameters:	1
Return type:	status
YANG file:	yuma-netconf.yang
Capabilities needed:	none
Capabilities optional:	:candidate :startup

Mandatory Parameters:

- **target**
  - type: container with 1 of N choice of leafs
  - This parameter specifies the name of the target database to be unlocked.
  - container contents: 1 of N:
    - **candidate**
      - type: empty
      - capabilities needed: :candidate
    - **running**
      - type: empty
      - capabilities needed: none
    - **startup**
      - type: empty
      - capabilities needed: startup

Optional Parameters:

- none

Returns:

- <ok/>

Possible Operation Errors:

- access denied
- no-access (database not locked)
- invalid-value (database not supported)

Example Request:

```
<?xml version="1.0" encoding="UTF-8"?>
<nc:rpc xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0"
   message-id="65">
   <nc:unlock>
      <nc:target>
         <nc:candidate/>
      </nc:target>
   </nc:unlock>
</nc:rpc>
```

Example Reply:

```
<?xml version="1.0" encoding="UTF-8"?>
<nc:rpc-reply xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0"
   message-id="65">
   <nc:ok/>
</nc:rpc-reply>
```

2.6.23 <VALIDATE>

The <validate> operation is used to perform the <commit> validation tests against a database or some in-line configuration data.

<validate> operation

Min parameters:	1
Max parameters:	1
Return type:	status
YANG file:	yuma-netconf.yang
Capabilities needed:	:validate
Capabilities optional:	:candidate :startup

Mandatory Parameters:

- **target**
  - type: container with 1 of N choice of leafs
  - This parameter specifies the name of the target database, or the in-line configuration data, that should be validated.
  - container contents: 1 of N:
    - **candidate**
      - type: empty
      - capabilities needed: :candidate
    - **running**
      - type: empty
      - capabilities needed: none
    - **startup**
      - type: empty
      - capabilities needed: startup
    - **config**
      - type anyxml
      - capabilities needed: none

Optional Parameters:

- none

Returns:

- <ok/>

Possible Operation Errors:

- access denied
- <commit> validation errors

Example Request:

```
<?xml version="1.0" encoding="UTF-8"?>
<nc:rpc xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0"
   message-id="76">
   <nc:validate>
      <nc:source>
         <nc:candidate/>
      </nc:source>
   </nc:validate>
</nc:rpc>
```

Example Reply:

```
<?xml version="1.0" encoding="UTF-8"?>
<nc:rpc-reply xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0"
   message-id="76">
   <nc:ok/>
</nc:rpc-reply>
```

2.7 Access Control

The **netconfd** access control data model is defined in **yuma-nacm.yang**.

The **nacm:secure** and **nacm:very-secure** extensions also affect access control. If present in the YANG definition, then the default behavior (when no rule is found) is not followed. Instead, the super user account must be used to allow default access.

There are 3 types of access control rules that can be defined:

1. module rule
2. RPC operation rule
3. database rule

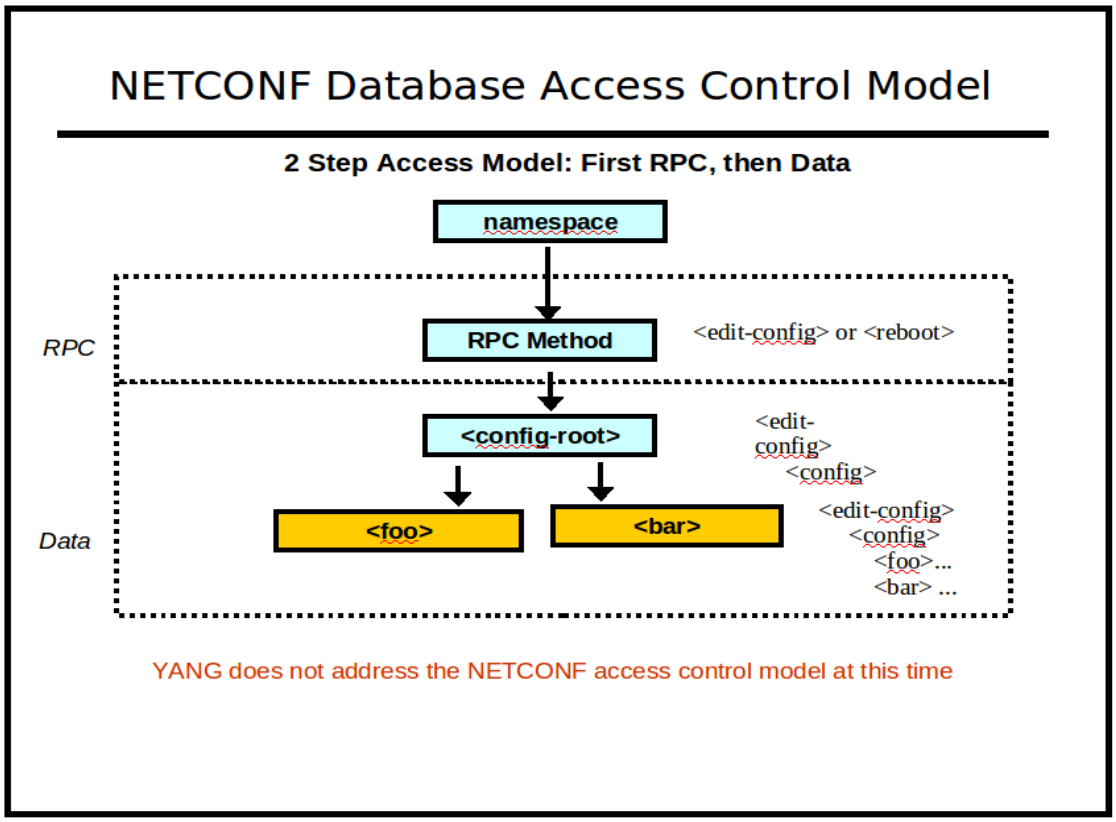
Rules apply to one or more groups.

Each group contains zero or more user names.

Database rules are applied top-down, at every level.

The user needs permission for the requested access (read or write) for all referenced nodes within the database. For example, if there was a leaf from module X that augments a container in module Y, the user would need permission to access the container from module Y, and then permission to access the leaf from module X.

The NACM data model can be used without any configuration at all. Refer to the section on access control modes for more details.



Normally, some configuration is required:

- groups: configure one or more administrative groups
- rules: configure one or more access control rules

The entire **/nacm** subtree is tagged as **nacm:very-secure**. By default, only the super-user account can read or write any of its contents. It is suggested that even read access to this data structure be controlled carefully at all times.

2.7.1 NACM Module Structure

The **/nacm** subtree consists of 3 read-only leafs and 2 containers:

- leaf **<enable-nacm>**: enable or disable access control enforcement
- leaf **<read-default>**: permit or deny read access when no rule found
- leaf **<write-default>**: permit or deny write access when no rule found
- leaf **<exec-default>**: permit or deny execute access when no rule found
- leaf **<denied-rpcs>**: read-only counter of denied RPC operation requests
- leaf **<denied-data-writes>**: read-only counter of denied database write requests
- container **<groups>**: need 1 or more **group** list entries in order for **rules** to have any effect
  - list **<group>**: principle that is assigned access privileges
    - leaf **<group-identity>**: group identifier string

        - leaf-list <**user-name**>: leaf-list of users that belong to the group
- container **<rules>**:
    - list **<module-rule>**: an access rule for an entire module namespace
        - leaf **<module-name>**: [key] name of the YANG module associated with the rule
        - leaf **<allowed-rights>**: [key] privileges granted to all groups associated with the rule
        - leaf-list **<allowed-group>**: leaf-list of one or more group identifiers that are associated with the rule
        - leaf **<comment>**: comment string for the rule (ignored by the server)
    - list **<rpc-rule>**: an access rule for one specific protocol operation, defined in a YANG rpc statement.
        - leaf **<rpc-module-name>**: [key] name of the YANG module associated with the rule
        - leaf <**rpc-name>:** [key] name of the RPC operation associated with the rule
        - leaf **<allowed-rights>**: [key] privileges granted to all groups associated with the rule
        - leaf-list **<allowed-group>**: leaf-list of one or more group identifiers that are associated with the rule
        - leaf **<comment>**: comment string for the rule (ignored by the server)
    - list **<data-rule>**: an user-ordered access rule for one or more database subtrees.
        - leaf **<name>**: [key] arbitrary name string for the rule
        - leaf **<path>**: XPath expression for the database instances which are associated with the rule
        - leaf **<allowed-rights>**: privileges granted to all groups associated with the rule
        - leaf-list **<allowed-group>**: leaf-list of one or more group identifiers that are associated with the rule
        - leaf **<comment**:> comment string for the rule (ignored by the server)
    - list **<notification-rule>**: an user-ordered access rule for one notification events.
        - leaf **<notification-module-name>**: [key] name of the YANG module associated with the rule
        - leaf <**notification-name>:** [key] name of the notification event type associated with the rule
        - leaf **<allowed-rights>**: [key] privileges granted to all groups associated with the rule
        - leaf **<allowed-rights>**: privileges granted to all groups associated with the rule
        - leaf-list **<allowed-group>**: leaf-list of one or more group identifiers that are associated with the rule
        - leaf **<comment**:> comment string for the rule (ignored by the server)

2.7.2 Users and Groups

Access rights in NACM are given to groups.

A group entry consists of a group identifier and a list of user names that are members of the group.

A group is named with is a YANG identity, which has a base of 'nacmRoot':

```
identity nacmGroups {
    description
       "Root of all NETCONF Administrative Groups";
}
```

There are 3 hard-wired group names that can be used:

- nacm:admin
- nacm:monitor
- nacm:guest

Any module can define an extension identity for the 'nacmGroups' base type, and use it as a group name. There are no special semantics associated with any particular group name.

```
import nacm { prefix nacm; }

identity mygroup {
    description
       "My special administrator's group.";
       base nacm:nacmGroups;
}
```

By default, there are no groups created. Each **/nacm/groups/group** entry must be created by the client. There is no user name table either. It is assumed that the operator will know which user names are valid within each managed device.

2.7.3 Creating New Groups

The <edit-config> operation can be used to create new group entries.

Each group is identified only by its <groupIdentity> leaf.

A user name can appear within the same group zero or one times.

A user name can appear in zero or more groups.

When a user is a member of multiple groups, all these groups will be used to match against rules, in a conceptual 'OR' expression. If any of these groups matches one of the <allowed-group> leaf-list nodes within one of the 3 rule types, then that rule will be the one that is used. Rules are always searched in the order they are entered, even the system-ordered lists.

The path to the group element is **/nacm:nacm/nacm:groups/nacm:group.**

The following <edit-config> example shows how a group can be defined. The group name is 'nacm:guest',and the users 'fred' and 'barney' are the initial members of this group.

```
<?xml version="1.0" encoding="UTF-8"?>
<nc:rpc xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0"
   message-id="3">
   <nc:edit-config>
      <nc:target>
         <nc:candidate/>
      </nc:target>
      <nc:config>
         <nacm:nacm xmlns:nacm="http://netconfcentral.org/ns/yuma-
nacm">
            <nacm:groups>
               <nacm:group  nc:operation="create">
                  <nacm:group-identity>nacm:guest</nacm:group-
identity>
                  <nacm:user-name>fred</nacm:user-name>
                  <nacm:user-name>barney</nacm:user-name>
               </nacm:group>
            </nacm:groups>
         </nacm:nacm>
      </nc:config>
   </nc:edit-config>
</nc:rpc>
```

2.7.4 Access Control Modes

The **--access-control** configuration parameter is used to globally enable or disable the access control system. It cannot be changed at run-time, or through a NETCONF session.

- **off**: no access control enforcement is done at all.
- **disabled**: all **nacm:secure** and **nacm:very-secure** tagged objects will require the super user account to access, but no other access control enforcement will be done.
- **permissive**: all **nacm:very-secure** objects will require super user account to read. No other read access enforcement will be done. Write and exec access will be checked.
- **enforcing**: all access control enforcement will be checked. This is the default behavior.

2.7.5 Permissions

There are 3 types of access permissions defined:

1. **read**: retrieval of any kind
2. **write**: modification of any kind
3. **exec**: right to invoke an RPC operation

The <allowed-rights> object in each of the 3 access control rule entries is a 'bits' leaf, which is allowed to contain any of these string tokens, or none of them to deny all access to a set of groups.

When a rule is found which matches the current request, the <allowed-rights> leaf will be used to grant permission or not. If the bit for the requested operation is present, then the request is permitted. If the bit is not present, then the request is denied.

2.7.6 Default Enforcement Behavior

Each access type has a default behavior if no rule is found:

- read default: permit
- write default: deny
- exec default: permit

These defaults can be changed by the server developer, by modifying the YANG definitions in yuma-nacm.yang.

2.7.7 Access Control Algorithm

The following logic represents the steps taken during access control enforcement.

Phase 1: RPC Operation Access

- Step 1) If the **--access-control** configuration parameter is set to 'off' then grant access and exit.
- Step 2) If the user name matches the **--superuser** configuration variable (or the default 'superuser' if not set), then grant access and exit.
- Step 3) If the RPC operation requested is the NETCONF <close-session> operation, then grant access and exit.
- Step 4) If the **--access-control** configuration parameter is set to 'disabled':
  - a) If the requested RPC operation is tagged as **nacm:secure** or **nacm:very-secure**, then deny access, otherwise grant access, and exit.
- Step 5) Retrieve all the NACM groups that this user is found within.
  - If there are no groups found:
    - a) If the requested RPC operation is tagged as **nacm:secure** or **nacm:very-secure**, then deny access and exit.
    - b) If the <exec-default> leaf is set to 'permit' then grant access, otherwise deny access, and exit.
  - If there are any groups found, then proceed to step 6.
- Step 6) Check if there are any **/nacm/rules** nodes configured.
  - If there are no rules configured, then if the <exec-default> leaf is set to 'permit' then grant access, otherwise deny access, and exit.
  - If there are some entries within the <rules> container, then proceed to step 7.
- Step 7) Check for any **/nacm/rules/rpc-rule** entries that contain an <allowed-group> with the same value as one of the groups found in step 5, and is also for the requested RPC operation. The first entry found will be used.
    - If an entry is found, then check its <allowed-rights> leaf for the 'exec' bit.
      - If the 'exec' bit is found, then grant access, otherwise deny access, and exit.
    - If a matching <rpc-rule> entry is not found, then proceed to step 8.
- Step 8) Check for any **/nacm/rules/module-rule** entries that contain an <allowed-group> with the same value as one of the groups found in step 5, and is also for same module as the requested RPC operation. The first entry found will be used.
    - If an entry is found, then check its <allowed-rights> leaf for the 'exec' bit.
      - If 'exec' bit is found, then grant access, otherwise deny access, and exit.

        - If a matching <module-rule> entry is not found, then proceed to step 9.
- Step 9) If the <exec-default> leaf is set to 'permit' then grant access, otherwise deny access.
    - Continue to phase 2a if any read access, or phase 2b if any write access to the NETCONF database contents is requested on behalf of the specific RPC operation.

Phase 2a: Database Read and Notification Access

- Step 10) If the user name matches the **--superuser** configuration variable (or the default 'superuser' if not set), then grant access and exit.
- Step 11) Check the access control mode:
    - a) If the **--access-control** configuration parameter is set to 'permissive'
        - If the requested object is tagged as **nacm:very-secure**, then deny access, otherwise grant access, and exit
    - b) Otherwise, this must be normal 'enforcing' mode, so proceed to step 12.
- Step 12) Make sure there are some groups and rules
    - a) If there were no groups found in step 5, or no rules found in step 6, then:
        - If the requested object is tagged as **nacm:very-secure**, then deny access, otherwise:
            - If the <read-default> is set to 'permit', then grant access, otherwise deny access, and exit.
    - b) If there are some groups and rules, then proceed to step 13 to start checking access control rules.
- Step 13) Check for any **/nacm/rules/data-rule** entries that contain a <path> expression that evaluates to an XPath node-set that contains the requested database node, and the <allowed-group> leaf-list contains an entry with the same value as one of the groups found in step 5. The first such entry found will be used.
        - If an entry is found, then check its <allowed-rights> leaf for the 'read' bit.
            - If the 'read' bit is found, then grant access, otherwise deny access, and exit.
        - If an entry is not found, then proceed to step 14.
- Step 14) Check for any **/nacm/rules/module-rule** entries that contain an <allowed-group> with the same value as one of the groups found in step 5, and is also for the same module as the requested database node. The first entry found will be used.
        - If an entry is found, then check its <allowed-rights> leaf for the 'read' bit.
            - If the 'read' bit is found, then grant access, otherwise deny access, and exit.
        - If an entry is not found, then proceed to step 15.
- Step 15) If the <read-default> leaf is set to 'permit' then grant access, otherwise deny access, and exit.

Phase 2b) Database Write Access

- Step 16) If the user name matches the **--superuser** configuration variable (or the default 'superuser' if not set), then grant access and exit.
- Step 17) Make sure there are some groups and rules
    - a) If there were no groups found in step 5, or no rules found in step 6, then:

        - If the requested object is tagged as **nacm:secure** or **nacm:very-secure**, then deny access, otherwise:
            - If the <write-default> is set to 'permit', then grant access, otherwise deny access, and exit.
    - b) If there are some groups and rules, then proceed to step 18 to start checking access control rules.
- Step 18) Check for any **/nacm/rules/data-rule** entries that contain a <path> expression that evaluates to an XPath node-set that contains the requested database node, and the <allowed-groups> leaf-list contains an entry with the same value as one of the groups found in step 5. The first such entry found will be used.
    - If an entry is found, then check its <allowed-rights> leaf for the 'write' bit.
        - If the 'write' bit is found, then grant access, otherwise deny access, and exit.
    - If an entry is not found, then proceed to step 19.
- Step 19) Check for any **/nacm/rules/module-mule** entries that contain an <allowed-group> with the same value as one of the groups found in step 5, and is also for the same module as the requested database node. The first entry found will be used.
    - If an entry is found, then check its <allowed-rights> leaf for the 'write' bit.
        - If the 'write' bit is found, then grant access, otherwise deny access, and exit.
    - If an entry is not found, then proceed to step 20.
- Step 20) If the <write-default> leaf is set to 'permit' then grant access, otherwise deny access, and exit.

2.7.8 Module Access Control Rules

The **/nacm/rules/module-rule** data structure is used to configure access for any object or RPC operation from a specific YANG module. If the module namespace URI is the same as the XML namespace used in the NETCONF PDU, then the module rule is considered a match.

Multiple instances can appear for a single module, as long as the <allowed-access> key leaf value is different in each entry. This allows different groups to get different access to the same module (e.g., read vs. read and write).

There is no way to move <module-rule> entries around, once they are created.

If a group appears in multiple entries for the same module name, then the first one encountered will be used. Entries are checked in the same order they are returned in a <get-config> reply message.

The following example shows an <edit-config> operation which creates 2 <moduleRule> entries, for the following configuration:

- the 'admin' group is allowed to read and write the NACM module.
- the 'monitor' group is allowed to read the NACM module.

```
<?xml version="1.0" encoding="UTF-8"?>
<nc:rpc xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0"
   message-id="4">
   <nc:edit-config>
      <nc:target>
         <nc:candidate/>
      </nc:target>
      <nc:config>
         <nacm:nacm xmlns:nacm="http://netconfcentral.org/ns/yuma-
nacm">
            <nacm:rules>
               <nacm:module-rule  nc:operation="create">
                  <nacm:module-name>nacm</nacm:module-name>
                  <nacm:allowed-rights>read write</nacm:allowed-
rights>
                  <nacm:allowed-group>nacm:admin</nacm:allowed-
group>
               </nacm:module-rule>
               <nacm:module-rule  nc:operation="create">
                  <nacm:module-name>nacm</nacm:module-name>
                  <nacm:allowed-rights>read</nacm:allowed-rights>
                  <nacm:allowed-group>nacm:monitor</nacm:allowed-
group>
               </nacm:module-rule>
            </nacm:rules>
         </nacm:nacm>
      </nc:config>
   </nc:edit-config>
</nc:rpc>
```

2.7.9 RPC Access Control Rules

The **/nacm/rules/rpc-rule** data structure is used to configure access for a specific RPC operation from a specific YANG module. If the module namespace URI for the <rpc-module-name> value is the same as the XML namespace used in the NETCONF PDU,and the <rpc--name> value is the same as the RPC method name, then the RPC rule is considered a match.

Multiple instances can appear for a single RPC operation, as long as the <allowed-access> key leaf value is different in each entry. This allows different groups to get different access to the same operation (e.g., exec vs. no access).

There is no way to move <rpc-rule> entries around, once they are created.

If a group appears in multiple entries for the same RPC operation, then the first one encountered will be used. Entries are checked in the same order they are returned in a <get-config> reply message.

If the 'read' or 'write' access bits are set in the <allowed-rights> key leaf, then they will be ignored. This will not cause an error, but it these bits have no effect within an RPC rule.

The following example shows an <edit-config> operation which creates 2 <module-rule> entries, for the following configuration:

- the 'admin' and 'monitor' groups are allowed to execute the <get-config> operation.
- the 'guest' group is **not** allowed to execute the <get-config> operation

```
<?xml version="1.0" encoding="UTF-8"?>
<nc:rpc xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0"
   message-id="5">
   <nc:edit-config>
      <nc:target>
         <nc:candidate/>
      </nc:target>
      <nc:config>
         <nacm:nacm xmlns:nacm="http://netconfcentral.org/ns/yuma-
nacm">
            <nacm:rules>
               <nacm:rpc-rule  nc:operation="create">
                  <nacm:rpc-module-name>netconf</nacm:rpc-module-
name>
                  <nacm:rpc--name>get-config</nacm:rpc--name>
                  <nacm:allowed-rights>exec</nacm:allowed-rights>
                  <nacm:allowed-group>nacm:admin</nacm:allowed-
group>
                  <nacm:allowed-group>nacm:monitor</nacm:allowed-
group>
               </nacm:rpc-rule>
               <nacm:rpc-rule  nc:operation="create">
                  <nacm:rpc-module-name>netconf</nacm:rpc-module-
name>
                  <nacm:rpc--name>get-config</nacm:rpc--name>
                  <nacm:allowed-rights/>
                  <nacm:allowed-group>nacm:guest</nacm:allowed-
group>
               </nacm:rpc-rule>
            </nacm:rules>
         </nacm:nacm>
      </nc:config>
   </nc:edit-config>
</nc:rpc>
```

2.7.10 Data Access Control Rules

The **/nacm/rules/data-rule** data structure is used to configure access for a specific set of database nodes (or notification payload nodes). If the requested node in contained within the node-set result of the <path> XPath path expression, then the data rule is considered a match.

Multiple instances of the same <path> expression (or equivalent expressions), can appear at any time, and in any order.

The <path> leaf is not allowed to contain an arbitrary XPath expression (at this time). Instead, an **ncx:schema-instance** string is allowed. This has the same syntax as a YANG instance-identifier built-in type, except that the key leaf predicates (e.g., [name='eth0']) are optional instead of mandatory. A missing key leaf predicate indicates that all instances of that key leaf are going to match the data rule.

This is a user-created list, and the key leaf is the arbitrary <name> field. Use the YANG 'insert' operation to add data rules in some order other than 'last'. Entries are checked in the same order they are returned in a <get-config> reply message.

If a group appears in multiple entries, then the first one that produces a result node-set with a matching node wiill be used.

If the 'exec' access bit is set in the <allowed-rights> key leaf, then it will be ignored. This will not cause an error, but it this bit has no effect within a data rule.

The following example shows an <edit-config> operation which creates 3 <data-rule> entries, for the following configuration:

- the 'admin' group is allowed to read or write all interfaces.
- the 'monitor' group is allowed to read the 'eth0' interface
- the 'guest' group is not allowed to read any interfaces information

```
<?xml version="1.0" encoding="UTF-8"?>
<nc:rpc xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0"
   message-id="3">
   <nc:edit-config>
      <nc:target>
         <nc:candidate/>
      </nc:target>
      <nc:default-operation>none</nc:default-operation>
      <nc:config>
         <nacm:nacm xmlns:nacm="http://netconfcentral.org/ns/yuma-
nacm">
            <nacm:rules>
               <nacm:data-rule  nc:operation="create">
                  <nacm:name>itf-1</nacm:name>
                  <nacm:path
                     xmlns:if="http://netconfcentral.org/ns/yuma-
interfaces">
                       /if:interfaces/if:interface
                  </nacm:path>
                  <nacm:allowed-rights>read write</nacm:allowed-
rights>
                  <nacm:allowed-group>nacm:admin</nacm:allowed-
group>
                  <nacm:comment>
                     let admin group read and write all interfaces
                  </nacm:comment>
               </nacm:data-rule>
               <nacm:data-rule  nc:operation="create">
                  <nacm:name>itf-2</nacm:name>
                  <nacm:path
                     xmlns:if="http://netconfcentral.org/ns/interfac
es">
                       /if:interfaces/if:interface[if:name='eth0']
                  </nacm:path>
                  <nacm:allowed-rights>read</nacm:allowed-rights>
                  <nacm:allowed-group>nacm:monitor</nacm:allowed-
group>
                  <nacm:comment>
                     let monitor group read interface 'eth0'
                  </nacm:comment>
               </nacm:data-rule>
               <nacm:data-rule  nc:operation="create">
                  <nacm:name>itf-3</nacm:name>
                  <nacm:path
                     xmlns:if="http://netconfcentral.org/ns/interfac
es">
```

```
                        /if:interfaces
                    </nacm:path>
                    <nacm:allowed-rights/>
                    <nacm:allowed-group>nacm:guest</nacm:allowed-
group>
                    <nacm:comment>
                       do not let guest group read any interfaces info
                    </nacm:comment>
                 </nacm:data-rule>
              </nacm:rules>
           </nacm:nacm>
        </nc:config>
     </nc:edit-config>
</nc:rpc>
```

2.8 Monitoring

The <get> and <get-config> operations are fully supported for retrieving data from the <candidate> and <running> configuration databases.

The <get-config> operation is not supported for the <startup> configuration.

The <copy-config> operation is only supported copying the <running> configuration to the <startup> configuration.

If the NACM access control policy denies permission to read a particular node, then that node is silently skipped in the output. No error or warning messages will be generated.

client applications should be prepared to receive XML subtrees that have been pruned by access control. The <data> element will always be present, so an empty <data/> element indicates that no data was returned, either because the <filter> did not match, or because access control pruned the requested nodes.

There are really five types of filters available for retrieval operations:

Filter Types

type	description
is_config()	Choose the <get>operation for all objects, or <get-config> for just config=true objects
is_default()	Set the <with-defaults> parameter to 'trim'
is_client_set()	Set the <with-defaults> parameter to 'explicit'
subtree filtering	Use <filter type="subtree"> // some-xml-subtree </filter> to retrieve portions of the <candidate> or <running> configurations.
XPath filtering	Use

	<filter type="xpath" select="expr"/> to retrieve portions of the <candidate> or <running> configurations.

2.8.1 Using Subtree Filters

The subtree filtering feature is fully supported.

The order of nodes within the <filter> element is not relevant. Data returned in the <rpc-reply> should follow the same top-level order as the request, but this should not be relied on to always be the case.

Duplicate or overlapping subtrees within the request will be combined in the output, so the common ancestor nodes are not duplicated in the reply.

XML namespaces are optional to use:

- If there is no namespace in affect for a filter component, or the 'NETCONF' namespace is in effect, the server will attempt to find any top-level data node which matches.
- Namespaces within descendant nodes of the <filter> node children are inherited from their parent. If none is in effect, then the first matching child node for the current parent node will be used.
- Invalid namespace errors for the <filter> element are suppressed in the server. An invalid namespace or unknown element is simply a 'no-match' condition.

For example, the following PDU would be valid, even though it is not technically valid XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<nc:rpc xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0"
   message-id="8">
   <nc:get>
      <nc:filter>
         <nacm>
           <rules/>
         </nacm>
      </nc:filter>
   </nc:get>
</nc:rpc>
```

Note that there is no default namespace in effect for the <nacm> subtree. However, the server will accept this filter as if the yuma-nacm.yang module namespace was properly declared.

Subtree filters can select specific list entries using content match nodes. The following example would return the entire contents of the <interface> entry for 'eth0':

```
<?xml version="1.0" encoding="UTF-8"?>
<nc:rpc xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0"
   message-id="9">
   <nc:get>
      <nc:filter type="subtree">
```

```
         <if:interfaces xmlns:if="http://netconfcentral.org/ns/yuma-
interfaces">
            <if:interface>
               <if:name>eth0</if:name>
            </if:interface>
         </if:interfaces>
      </nc:filter>
   </nc:get>
</nc:rpc>
```

To retrieve only specific nodes (such as counters) from a single list entry, use select nodes for the desired counter(s), and include a content match node for each key leaf. A missing key leaf will match any entry for that key.

The following example request shows how just the <inBytes> and <outBytes> counters could be retrieved from the <interface> entry for 'eth0'.

Example Request:

```
<?xml version="1.0" encoding="UTF-8"?>
<nc:rpc xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0"
   message-id="10">
   <nc:get>
      <nc:filter type="subtree">
         <if:interfaces xmlns:if="http://netconfcentral.org/ns/yuma-
interfaces">
            <if:interface>
               <if:name>eth0</if:name>
               <if:counters>
                  <if:inBytes/>
                  <if:outBytes/>
               </if:counters>
            </if:interface>
         </if:interfaces>
      </nc:filter>
   </nc:get>
</nc:rpc>
```

Example Reply:

```
<?xml version="1.0" encoding="UTF-8"?>
<nc:rpc-reply xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0"
   message-id="10">
   <nc:data>
      <if:interfaces
xmlns:if="http://netconfcentral.org/ns/interfaces">
         <if:interface>
            <if:name>eth0</if:name>
            <if:counters>
               <if:inBytes>290046042</if:inBytes>
               <if:outBytes>112808406</if:outBytes>
            </if:counters>
         </if:interface>
      </if:interfaces>
   </nc:data>
</nc:rpc-reply>
```

2.8.2 Using XPath Filters

The :xpath capability is fully supported, including the YANG extensions to this capability.

The XPath 2.0 rule for default XML namespace behavior is used, not XPath 1.0 rules, as specified by the YANG language. This means that any module with a node with the same local-name, in the same position in the schema tree, will match a missing XML prefix. This allows much simpler specification of XPath filters, but it may match more nodes than intended. Remember that any nodes added via an external YANG augment statement may have the same local-name, even though they are bound to a different XML namespace.

If the XPath expression does not return a node-set result, then the empty <data/> element will be returned in the <rpc-reply>.

If no nodes in the node-set result exist in the specified target database, then an empty <data/> element will be returned in the <rpc-reply>.

If a node in the result node-set matches a node in the target database, then it is included in the <rpc-reply>,

If a node selected for retrieval are contained within a YANG list node, then all the key leaf nodes for the specific list entry will be returned in the response.

The powerful '//' operator (equivalent to "descendant-or-self::node()") can be used to construct really simple XPath expressions.

The following example shows how a simple filter like '//name' will return nodes from all over the database, yet they can all be fully identified because the path from root is part of the response data.

Example Request:

- xget //name

```
<nc:rpc xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0"
   message-id="43">
   <nc:get>
      <nc:filter type="xpath" select="//name"/>
   </nc:get>
</nc:rpc>]]>]]>
```

Example Reply:

```
<?xml version="1.0" encoding="UTF-8"?>
<nc:rpc-reply xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0"
   message-id="43">
   <nc:data>
      <nacm:nacm xmlns:nacm="http://netconfcentral.org/ns/nacm">
         <nacm:rules>
            <nacm:data-rule>
               <nacm:name>nacm-tree</nacm:name>
            </nacm:data-rule>
            <nacm:data-rule>
               <nacm:name>itf-1</nacm:name>
            </nacm:data-rule>
         </nacm:rules>
      </nacm:nacm>
      <t:xpath.1 xmlns:t="http://netconfcentral.org/ns/test">
         <t:name>barney</t:name>
      </t:xpath.1>
      <if:interfaces
xmlns:if="http://netconfcentral.org/ns/interfaces">
         <if:interface>
            <if:name>lo</if:name>
         </if:interface>
         <if:interface>
            <if:name>eth0</if:name>
         </if:interface>
         <if:interface>
            <if:name>virbr0</if:name>
```

```
         </if:interface>
         <if:interface>
            <if:name>pan0</if:name>
         </if:interface>
      </if:interfaces>
      <ns:netconf-state
xmlns:ns="urn:ietf:params:xml:ns:netconf:monitoring">
         <ns:datastores>
            <ns:datastore>
               <ns:name>
                  <ns:candidate/>
               </ns:name>
            </ns:datastore>
            <ns:datastore>
               <ns:name>
                  <ns:running/>
               </ns:name>
            </ns:datastore>
            <ns:datastore>
               <ns:name>
                  <ns:startup/>
               </ns:name>
            </ns:datastore>
         </ns:datastores>
      </ns:netconf-state>
      <manageEvent:netconf
xmlns:manageEvent="urn:ietf:params:xml:ns:netmod:notification">
         <manageEvent:streams>
            <manageEvent:stream>
               <manageEvent:name>NETCONF</manageEvent:name>
            </manageEvent:stream>
         </manageEvent:streams>
      </manageEvent:netconf>
   </nc:data>
</nc:rpc-reply>
```

In order to refine the previous filter to select nodes from just one module, the use the XML prefix in the node identifier. The example below selects only the <name> nodes from the interfaces module.

Example Request:

- xget //if:name

```
<?xml version="1.0" encoding="UTF-8"?>
<nc:rpc xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0"
   message-id="44">
   <nc:get>
      <nc:filter type="xpath"
         xmlns:if="http://netconfcentral.org/ns/interfaces"
         select="//if:name"/>
   </nc:get>
</nc:rpc>
```

Example Reply:

```
<?xml version="1.0" encoding="UTF-8"?>
<nc:rpc-reply xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0"
   message-id="44">
   <nc:data>
      <if:interfaces
xmlns:if="http://netconfcentral.org/ns/interfaces">
         <if:interface>
            <if:name>lo</if:name>
         </if:interface>
         <if:interface>
            <if:name>eth0</if:name>
         </if:interface>
         <if:interface>
            <if:name>virbr0</if:name>
         </if:interface>
         <if:interface>
            <if:name>pan0</if:name>
         </if:interface>
      </if:interfaces>
   </nc:data>
</nc:rpc-reply>
```

2.9 Notifications

The **netconfd** server supports all the capabilities of RFC 5277, and the notification monitoring portion of the **ietf-netconf-monitoring.yang** data model. There are also some proprietary notifications defined in **yuma-system.yang**.

2.9.1 Subscriptions

The <create-subscription> operation is used to start receiving notification.

It can be used in 4 different modes:

- Get all or some of the stored notifications.
  - <startTime> and <stopTime> parameters used; The stop time is in the past.
- Get all or some of the stored notifications, then receive live notifications until some point in the future.
  - <startTime> and <stopTime> parameters used; The stop time is in the future.
- Get all or some of the stored notifications, then start receiving live notifications until the session is terminated.
  - <startTime> parameter used, but <stopTime> parameter is not used
- Start receiving live notifications until the session is terminated
  - neither <startTime> or <startTime> are used

Once a subscription is started, notifications may start arriving, after the <rpc-reply> for the <create-subscription> operation is sent.

If the <startTime> parameter is used, then zero or more stored notifications will be returned, followed by the <replayComplete> notification.

If the <stopTime> parameter is also used, then the <notificationComplete> notification will be sent when this stop time has passed. After that, no more notifications will be sent to the session, and the subscription is terminated. After this point, another subscription could be started.

Only one subscription can be active on a session at a time. There is no way to terminate a subscription, other than to close the session.

2.9.2 Notification Log

Each system event is saved to the notification replay buffer.

The <replayComplete> and <notificationComplete> notifications are not saved to this buffer because they are subscription-specific events, and not system events.

The size of the replay buffer is controlled by the **--eventlog-size** configuration parameter.

The default size is 1000 events.

The oldest event will be deleted when a new event is added, when this limit is reached.

If **--eventlog-size** is set to zero, then there will be no replayed notifications available, and the <replayComplete> notification will be sent right away, if <startTIme> is present.

Each event in the replay buffer is assigned a sequential sequence ID, starting from 1.

The <sequence-id> leaf is an unsigned 32-bit integer, which is added to the <notification> element, after the event element. This sequence can be used to debug filters by comparing the sequence IDs of the notifications that were delivered against the expected sequence IDs.

2.9.3 Using Notification Filters

A notification filter is different than a <get> or <get-config> filter.

Instead of selecting sub-trees of the specified database, it is treated as a boolean expression. If the filter matches the content in the notification, then the notification is sent to that subscription. If the filter does not match the content, then the notification is not sent to that subscription.

A filter match for notification purposes means that the filter is conceptually applied, as if it were a <get> operation, and if any nodes are selected (non-empty result node-set), then the filter is a match. If no content is selected (empty result node-set), then the filter is not a match.

The first node that can appear in the filter is the event type. The <eventTime> and <sequence-id> nodes are siblings of the event type element, so they cannot be used in a notification filter.

2.9.4 <notification> Element

The notification element contains 2 or 3 child elements, in this order:

1. **eventTime**: timestamp for the event. The namespace URI for this element is "urn:ietf:params:xml:ns:netconf:notification:1.0"
2. **eventType**: The real name will be the name of the YANG notification, such as 'sysStartup'. The contents of this element will depend on the YANG notification definition. The namespace URI for this element will be different for every event type. It will be the same value as the YANG namespace statement in the module that defines the notification statement for the particular event type.
3. **sequence-id**: The system event sequence ID. Session or subscription specific events, such as 'replayComplete' and 'notficationComplete' do not have this element. The namespace URI for this element is "http://netconfcentral.org/ns/system".

2.9.5 <replayComplete> Event

The <replayComplete> event is generated on a subscription that requested notification replay (by supplying the <startTime> parameter).

This event type cannot be filtered out. The server will always attempt to deliver this notification event type when it is generated.

<replayComplete> notification

Description:	Buffered notification delivery has ended for a subscription
Min parameters:	0
Max parameters:	0
YANG file:	nc-notifications.yang

Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<ncEvent:notification
     xmlns:ncEvent="urn:ietf:params:xml:ns:netconf:notification:1.0"
>
   <ncEvent:eventTime>2009-07-29T17:21:37Z</ncEvent:eventTime>
   <manageEvent:replayComplete
      xmlns:manageEvent="urn:ietf:params:xml:ns:netmod:notification"
/>
</ncEvent:notification>
```

2.9.6 <notificationComplete> Event

The <notificationComplete> event is generated on a subscription that requested notification replay, and requested that the notification delivery stop (i.e., terminate subscription), after a certain time, using the <stopTime> parameter.

This event type cannot be filtered out. The server will always attempt to deliver this notification event type when it is generated.

<notificationComplete> notification

Description:	All notification delivery has ended, and the subscription is terminated.
Min parameters:	0
Max parameters:	0
YANG file:	nc-notifications.yang

Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<ncEvent:notification
     xmlns:ncEvent="urn:ietf:params:xml:ns:netconf:notification:1.0"
>
   <ncEvent:eventTime>2009-07-29T17:31:22Z</ncEvent:eventTime>
   <manageEvent:notificationComplete
      xmlns:manageEvent="urn:ietf:params:xml:ns:netmod:notification"
/>
</ncEvent:notification>
```

2.9.7 <sysStartup> Event

The <sysStartup> event is the first notification generated when the server starts or restarts. It contains the startup file source (if any) and lists any <rpc-error> contents that were detected at boot-time, during the copying of the startup configuration into the running configuration.

<sysStartup> notification

Description:	The **netconfd** server has started
Min parameters:	0
Max parameters:	2
YANG file:	yuma-system.yang

Parameters:

- **startupSource**
  - type: string
  - usage: optional (will not be present if **--no-startup** was present)
  - This parameter identifies the local file specification associated with the source of the startup configuration contents.
- **bootError**
  - type: list (same structure as <rpc-error> element)
  - usage: optional (will only be present if errors were recorded during boot)
  - There will be one entry for each <rpc-error> encountered during the load config operation. The <rpc-error> fields are used directly.
  - There is no particular order, so no key is defined.
  - All fields except <error-info> will be present from the original <rpc-error> that was generated during the boot process.

Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<ncEvent:notification
xmlns:ncEvent="urn:ietf:params:xml:ns:netconf:notification:1.0">
   <ncEvent:eventTime>2009-07-29T17:21:13Z</ncEvent:eventTime>
   <sys:sysStartup xmlns:sys="http://netconfcentral.org/ns/system">
      <sys:startupSource>
         /home/andy/swdev/yuma/trunk/netconf/data/startup-cfg.xml
      </sys:startupSource>
   </sys:sysStartup>
   <sys:sequence-id
      xmlns:sys="http://netconfcentral.org/ns/system">1
  </sys:sequence-id>
</ncEvent:notification>
```

2.9.8 <sysSessionStart> Event

The <sysSessionStart> notification is generated when a NETCONF session is started.

The username, remote address, and session ID that was assigned are returned in the event payload.

<sysSessionStart> notification

Description:	A new NETCONF session has started.
Min parameters:	3
Max parameters:	3
YANG file:	yuma-system.yang

Parameters:

- **userName**
  - type: string
  - usage: mandatory
  - This parameter identifies the SSH user name that is associated with the session.
- **sessionId**
  - type: uint32 (range 1 to max)
  - usage: mandatory
  - This parameter identifies the NETCONF session ID assigned to the session.
- **remoteHost**
  - type: inet:ip-address
  - usage: mandatory
  - This parameter identifies the remote host IP address that is associated with the session.

Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<ncEvent:notification
xmlns:ncEvent="urn:ietf:params:xml:ns:netconf:notification:1.0">
   <ncEvent:eventTime>2009-07-29T21:53:04Z</ncEvent:eventTime>
   <sys:sysSessionStart
xmlns:sys="http://netconfcentral.org/ns/system">
      <sys:userName>andy</sys:userName>
      <sys:sessionId>2</sys:sessionId>
      <sys:remoteHost>192.168.0.6</sys:remoteHost>
   </sys:sysSessionStart>
   <sys:sequence-id
      xmlns:sys="http://netconfcentral.org/ns/system">4
   </sys:sequence-id>
</ncEvent:notification>
```

2.9.9 <sysSessionEnd> Event

The <sysSessionEnd> notification is generated when a NETCONF session is terminated.

The username, remote address, and session ID that was assigned are returned in the event payload. The termination reason is also included.

If the session was terminated before it properly started, it is possible that there will not be a <sysSessionStart> notification event to match the <sysSessionEnd> event. For example, if the initial SSH connection setup fails before the <hello> message is processed, then only a <sysSessionEnd> notification event will be generated. In this case, the user name and other session information may not be available.

<sysSessionEnd> notification

Description:	A NETCONF session has terminated.
Min parameters:	1
Max parameters:	5
YANG file:	yuma-system.yang

Parameters:

- **userName**
  - type: string
  - usage: mandatory
  - This parameter identifies the SSH user name that is associated with the session.
- **sessionId**
  - type: uint32 (range 1 to max)
  - usage: mandatory
  - This parameter identifies the NETCONF session ID assigned to the session.
- **remoteHost**
  - type: inet:ip-address

- usage: mandatory
- This parameter identifies the remote host IP address that is associated with the session.

- **killedBy**
  - type: uint32 (range 1 to max)
  - usage: optional (will only be present if the terminationReason leaf is equal to 'killed'.
  - This parameter identifies the session number of the session that issued the <kill-session> operation.
- **terminationReason**
  - type: enumeration (closed, killed, dropped, timeout, bad-start, bad-hello, other)
  - usage: mandatory
  - This parameter indicates why the session was terminated.

Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<ncEvent:notification
xmlns:ncEvent="urn:ietf:params:xml:ns:netconf:notification:1.0">
   <ncEvent:eventTime>2009-07-29T21:53:12Z</ncEvent:eventTime>
   <sys:sysSessionEnd
xmlns:sys="http://netconfcentral.org/ns/system">
      <sys:userName>andy</sys:userName>
      <sys:sessionId>2</sys:sessionId>
      <sys:remoteHost>192.168.0.6</sys:remoteHost>
      <sys:terminationReason>closed</sys:terminationReason>
   </sys:sysSessionEnd>
   <sys:sequence-id
         xmlns:sys="http://netconfcentral.org/ns/system">5
   </sys:sequence-id>
</ncEvent:notification>
```

2.9.10 <sysConfigChange> Event

The <sysConfigChange> notification is generated when the <running> configuration database is altered by a NETCONF session.

If the **:candidate** capability is supported, then this event is generated when the <commit> operation completes.

If the **:writable-running** capability is supported instead, then this even is generated when the <edit-config> operation completes.

The user name, remote address, and session ID that made the change are reported.

A summary of the changes that were made is also included in the event payload.

If multiple changes are made at once, then one <sysConfigChange> event will be generated for each change. There is no significance to the order that these events are generated.

<sysConfigChange> notification

Description:	The <running> configuration has been changed by a NETCONF

	session.
Min parameters:	4
Max parameters:	4
YANG file:	yuma-system.yang

Parameters:

- **userName**
  - type: string
  - usage: mandatory
  - This parameter identifies the SSH user name that is associated with the session.
- **sessionId**
  - type: uint32 (range 1 to max)
  - usage: mandatory
  - This parameter identifies the NETCONF session ID assigned to the session.
- **remoteHost**
  - type: inet:ip-address
  - usage: mandatory
  - This parameter identifies the remote host IP address that is associated with the session.
- **edit**
  - list (with no key) of edit operations performed, 1 entry for each edit:
    - **target**
      - type: instance-identifier
      - usage: mandatory
      - This parameter contains the absolute path expression of the database object node that was modified.
    - **operation**
      - type: enumeration (merge, replace, create, delete)
      - usage: mandatory
      - This parameter identifies the nc:operation that was performed on the target node in the database.

Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<ncEvent:notification
xmlns:ncEvent="urn:ietf:params:xml:ns:netconf:notification:1.0">
   <ncEvent:eventTime>2009-07-29T22:21:18Z</ncEvent:eventTime>
   <sys:sysConfigChange
xmlns:sys="http://netconfcentral.org/ns/system">
      <sys:userName>andy</sys:userName>
      <sys:sessionId>3</sys:sessionId>
      <sys:remoteHost>192.168.0.6</sys:remoteHost>
      <sys:edit>
         <sys:target>
         /nd:config/nacm:nacm/nacm:groups/nacm:group[nacm:groupIdent
ity=nacm:admin]
         </sys:target>
         <sys:operation>create</sys:operation>
      </sys:edit>
   </sys:sysConfigChange>
   <sys:sequence-id
      xmlns:sys="http://netconfcentral.org/ns/system">7</sys:sequenc
e-id>
</ncEvent:notification>
```

2.9.11 <sysCapabilityChange> Event

The <sysCapabilityChange> notification is generated when the set of active capabilities for the server has changed.

The most common way this notification is generated is after the <load> operation has been used to add a new YANG module to the system.

It is possible that this notification will be generated for removal of capabilities. However, at this time, there are no NETCONF capabilities that can be removed from the running system.

The <added-capability> leaf list will contain the capability URI for each new capability that has just been added.

The <deleted-capability> leaf list will contain the capability URI for each existing capability that has just been deleted.

The <changedBy> container will identity whether the server or a NETCONF session caused the capability change.

If the change was made by the server, then this container will have an empty leaf named <server>.

If the change was made by a NETCONF session, the user name, remote address, and session ID for the session that caused the change are reported.

If multiple changes are made at once, then one <sysCapabilityChange> event will be generated for all the changes. There will be multiple instances of the <added-capability> or <deleted-capability> leaf-list elements in this case.

When this notification is generated, the i**etf-netconf-monitoring** data model <capabilities> data structure is updated to reflect the changes.

<sysCapabilityChange> notification

Description:	The set of currently active

	<capability> URIs has changed
Min parameters:	2
Max parameters:	5
YANG file:	yuma-system.yang

Parameters:

- choice changed-by (server or by-user)
  - **server**
    - type: empty
    - usage: mandatory
    - If this empty leaf is present, then the server caused the capability change.
  - case by-user (if a NETCONF session caused the capability change)
    - **userName**
      - type: string
      - usage: mandatory
      - This parameter identifies the SSH user name that is associated with the session.
    - **sessionId**
      - type: uint32 (range 1 to max)
      - usage: mandatory
      - This parameter identifies the NETCONF session ID assigned to the session.
    - **remoteHost**
      - type: inet:ip-address
      - usage: mandatory
      - This parameter identifies the remote host IP address that is associated with the session.
- **added-capability**
  - type:leaf-list of capability URI strings
  - usage: optional
  - This parameter contains one entry for each capability that was just added
- **deleted-capability**
  - type:leaf-list of capability URI strings
  - usage: optional
  - This parameter contains one entry for each capability that was just deleted.

Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<ncEvent:notification
xmlns:ncEvent="urn:ietf:params:xml:ns:netconf:notification:1.0">
   <ncEvent:eventTime>2009-07-29T23:03:06Z</ncEvent:eventTime>
   <sys:sysCapabilityChange
xmlns:sys="http://netconfcentral.org/ns/system">
      <sys:changed-by>
         <sys:userName>andy</sys:userName>
         <sys:sessionId>3</sys:sessionId>
         <sys:remoteHost>192.168.0.61</sys:remoteHost>
      </sys:changed-by>
      <sys:added-capability>
         http://netconfcentral.org/ns/toaster?
module=toaster&amp;revision=2009-06-23&amp;features=clock
      </sys:added-capability>
   </sys:sysCapabilityChange>
   <sys:sequence-id
      xmlns:sys="http://netconfcentral.org/ns/system">8
   </sys:sequence-id>
</ncEvent:notification>
```

2.9.12 <sysConfirmedCommit> Event

The <sysConfirmedCommit> notification is generated when the state of the confirmed-commit procedure has changed.

The confirmed-commit procedure is started when a <commit> operation with a <confirmed/> parameter is executed.

A <sysConfirmedCommit> notification is generated several times for a single confirmed-commit procedure. One or more of the following sub-events will be generated:

- **start**: A confirmed-commit procedure has started.
  - Sent once when the first <commit> operation is executed.
  - This event starts the confirmed-commit procedure.
  - If the <candidate> database is not altered, then the confirmed commit procedure will be skipped.
- **cancel**: A confirmed-commit procedure has been canceled
  - Sent only if the original session is terminated.
  - This event terminates the confirmed-commit procedure.
- **timeout**: A confirmed-commit procedure has timed out.
  - Sent only if the confirm-timeout interval expires.
  - This event terminates the confirmed-commit procedure.
- **extend**: A confirmed-commit procedure has been extended.
  - Sent if the 2nd to N-1th <confirm> operation contains a <confirmed/> parameter.
  - This event restarts the confirm-timeout interval, but does not reset the backup database.
  - Any new changes in the <candidate> database will be committed.
- **complete**: A confirmed-commit procedure has completed.

- Sent if the 2nd to Nth <commit> operation is executed before the confirm-timeout interval expires.
- This event terminates the confirmed-commit procedure.

<sysConfirmedCommit> notification

Description:	The state of the confirmed-commit procedure has changed.
Min parameters:	1
Max parameters:	4
YANG file:	yuma-system.yang

Parameters:

- **userName**
  - type: string
  - usage: mandatory
  - This parameter identifies the SSH user name that is associated with the session that caused the confirmed-commit procedure state change.
- **sessionId**
  - type: uint32 (range 1 to max)
  - usage: mandatory
  - This parameter identifies the NETCONF session ID assigned to the session.
- **remoteHost**
  - type: inet:ip-address
  - usage: mandatory
  - This parameter identifies the remote host IP address that is associated with the session.
- **confirmEvent**
  - type: enumeration (start, cancel, timeout, extend, complete)
  - usage: mandatory
  - This parameter indicates why the confirmed-commit procedure changed state.

Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<ncEvent:notification
xmlns:ncEvent="urn:ietf:params:xml:ns:netconf:notification:1.0">
   <ncEvent:eventTime>2009-08-29T23:03:06Z</ncEvent:eventTime>
   <sys:sysConfirmedCommit
xmlns:sys="http://netconfcentral.org/ns/system">
      <sys:userName>andy</sys:userName>
      <sys:sessionId>3</sys:sessionId>
      <sys:remoteHost>192.168.0.61</sys:remoteHost>
      <sys:confirmEvent>start</sys:confirmEvent>
   </sys:sysConfirmedCommit>
   <sys:sequence-id
```

```
      xmlns:sys="http://netconfcentral.org/ns/system">9
   </sys:sequence-id>
</ncEvent:notification>
```

3 CLI Reference

The **netconfd** program uses command line interface (CLI) parameters to control program behavior.

The following sections document all the Yuma CLI parameters relevant to this program, in alphabetical order.

3.1 --access-control

The **--access-control** parameter specifies how access control is enforced in the **netconfd** program.

It is an enumeration with the following values:

- **enforcing**: All configured access control rules will be enforced.
- **permissive**: All configured access control rules will be enforced for write and execute requests. All read requests will be allowed, unless the requested object contains the **nacm:very-secure** extension. In that case, all configured access control rules will be enforced, and no default access will be allowed.
- **disabled**: All read, write, and execute requests will be allowed, unless the object contains the **nacm:secure** or **nacm:very-secure** extension.
  - If the **nacm:secure** extension is in effect, then all configured access control rules will be enforced for write and execute requests.
  - If the **nacm:very-secure** extension is in effect, then all configured access control rules will be enforced for all requests. Use this mode with caution.
- **off**: All access control enforcement is disabled. Use this mode with extreme caution.

--access-control parameter

Syntax	enumeration: **enforcing** **permissive** **disabled** **off**
Default:	enforcing
Min Allowed:	0
Max Allowed:	1
Supported by:	netconfd
Example:	`netconfd \` `  --access-control=permissive`

3.2 --config

The **--config** parameter specifies the name of a Yuma configuration file that contains more parameters to process, in addition to the CLI parameters.

Refer to the 'Configuration Files' section for details on the format of this file.

--config parameter

Syntax	string: complete file specification of the text file to parse for more parameters.
Default:	/etc/yuma/<program-name>.conf
Min Allowed:	0
Max Allowed:	1
Supported by:	netconfd yangcli yangdiff yangdump
Example:	`netconfd \` `  --config=~/testconf.conf`

3.3 --datapath

The **--datapath** parameter specifies the directory search path to use while searching for data files. It consists of a colon (':') separated list of path specifications, commonly found in Unix, such as the **$PATH** environment variable.

This parameter overrides the **$YUMA_DATAPATH** environment variable, if it is present.

--datapath parameter

Syntax	string: list of directory specifications
Default:	**$YUMA_DATAPATH** environment variable
Min Allowed:	0
Max Allowed:	1
Supported by:	netconfd yangcli yangdiff yangdump
Example:	`netconfd \` `  --datapath="~/work2:~/data"`

3.4 --default-style

The **--default-style** parameter specifies the way leafs with default values are returned from the server for data retrieval operations. This setting will be used as the default behavior when processing the operations that support the <with-defaults> extension, and no value is provided.

The values and their meanings are defined in **ietf-with-defaults.yang**. Here is a summary:

- **report-all**: Report all nodes as the default behavior. The will be the behavior used if this parameter is not specified.

- **trim**: Report only nodes that do not have the server-assigned value, as the default behavior. This includes all leafs with a YANG default and any other node created by the server.
- **explicit**: Report only nodes that have been set by client action, as the default behavior. Any node created via the <startup> configuration at boot-time is considered to be a client-created node. It does not matter what the actual values are, with respect to YANG defaults or server-supplied defaults. Any nodes created by the server are skipped.

--default-style parameter

Syntax	enumeration: **report-all** **trim** **explicit**
Default:	report-all
Min Allowed:	0
Max Allowed:	1
Supported by:	netconfd
Example:	`netconfd \` `  --default-style=trim`

3.5 --deviation

The **--deviation** parameter is a leaf-list of modules that should be loaded automatically when the program starts, as a deviation module. In this mode, only the deviation statements are parsed and then made available later when the module that contains the objects being deviated is parsed.

The deviations must be known to the parser before the target module is parsed.

This parameter is used to identify any modules that have deviation statements for the set of modules being parsed (e.g., **--module** and **--subtree** parameters).

A module can be listed with both the **--module** and **--deviation** parameters, but that is not needed unless the module contains external deviations. If the module only contains deviations for objects in the same module, then the **--deviation** parameter does not need to be used.

The program will attempt to load each module in deviation parsing mode, in the order the parameters are entered.

For the **netconfd** program, If any modules have fatal errors then the program will terminate.

For the **yangdump** and **yangcli** programs, each module will be processed as requested.

--deviation parameter

Syntax	module name or filespec
Default:	none
Min Allowed:	0
Max Allowed:	unlimited
Supported by:	netconfd

	yangcli yangdump
Example:	`netconfd \` `  --deviation=test1_deviations`

3.6 --eventlog-size

The **--eventlog-size** parameter controls the maximum number of events that will be stored in the notification replay buffer by the **netconfd** server.

If set to 0, then notification replay will be disabled, meaning that all requests for replay subscriptions will cause the <replayComplete> event to be sent right away, since there are no stored notifications.

The server will delete the oldest entry, when this limit is reached and a new event is added to the replay buffer.

No memory is actually set aside for the notification replay buffer, so memory limits may be reached before the maximum number of events is actually stored, at any given time.

--eventlog-size parameter

Syntax	uint32
Default:	1000
Min Allowed:	0
Max Allowed:	1
Supported by:	netconfd
Example:	`netconfd --eventlog-size=20000`

3.7 --feature-disable

The **--feature-disable** parameter directs all programs to disable a specific feature.

This parameter is a formatted string containing a module name, followed by a colon ':', followed by a feature name, e.g.,

```
test:feature1
```

It is an error if a **--feature-enable** and **--feature-disable** parameter specify the same feature.

Parameters for unknown features will be ignored.

--feature-disable parameter

Syntax	formatted string (module:feature

Default:	none
Min Allowed:	0
Max Allowed:	unlimited
Supported by:	yangcli yangdiff yangdump netconfd
Example:	`netconfd\` `  --feature-disable=test:feature1`

3.8 --feature-enable

The **--feature-enable** parameter directs all programs to enable a specific feature.

This parameter is a formatted string containing a module name, followed by a colon ':', followed by a feature name, e.g.,

```
    test:feature1
```

It is an error if a **--feature-disable** and **--feature-enable** parameter specify the same feature.

Parameters for unknown features will be ignored.

--feature-enable parameter

Syntax	formatted string (module:feature
Default:	none
Min Allowed:	0
Max Allowed:	unlimited
Supported by:	yangcli yangdiff yangdump netconfd
Example:	`netconfd` `  --feature-enable=test:feature1`

3.9 --feature-enable-default

The **--feature-enable-default** parameter controls how **yangdump** will generate C code for YANG features by default.

If 'true', then by default, features will be enabled.

If 'false', then by default, features will be disabled.

If a **--feature-enable** or **--feature-disable** parameter is present for a specific feature, then this parameter will be ignored for that feature.

--feature-enable-default parameter

Syntax	boolean (true or false)
Default:	TRUE
Min Allowed:	0
Max Allowed:	1
Supported by:	yangcli yangdiff yangdump netconfd
Example:	`netconfd \` `  --feature-enable-default=false`

3.10 --hello-timeout

The **--hello-timeout** parameter controls the maximum number of seconds that the **netconfd** server will wait for a <hello> PDU, for each session.

If set to 0, then hello state timeouts will be disabled, meaning that no sessions will be deleted while waiting for a <hello> PDU.

It is strongly suggested that this parameter not be disabled, since a denial-of-service attack will be possible if sessions are allowed to remain in the 'hello wait' state forever. A finite number of SSH and NETCONF sessions are supported, so if an attacker simply opened lots of SSH connections to the netconf subsystem, the server would quickly run out of available sessions.

Sessions cannot be deleted manually via <kill-session> operation if no new sessions are being allocated by the server.

--hello-timeout parameter

Syntax	uint32; 0 == disabled; range 10 .. 3600 seconds (1 hour max)
Default:	600 (10 minutes)
Min Allowed:	0
Max Allowed:	1
Supported by:	netconfd
Example:	`netconfd --hello-timeout=300`

3.11 --help

The **--help** parameter causes program help text to be printed, and then the program will exit instead of running as normal.

This parameter can be combined with the **--help-mode** parameter to control the verbosity of the help text. Use **--brief** for less, and **--full** for more than the normal verbosity.

This parameter can be combined with the **--version** parameter in all programs. It can also be combined with the **--show-errors** parameter in **yangdump**.

The program configuration parameters will be displayed in alphabetical order, not in schema order.

--help parameter

Syntax	empty
Default:	none
Min Allowed:	0
Max Allowed:	1
Supported by:	netconfd yangcli yangdiff yangdump
Example:	`netconfd --help`

3.12 --help-mode

The **--help-mode** parameter is used to control the amount of detail printed when help text is requested in some command. It is always used with another command, and makes no sense by itself. It is ignored unless used with the **--help** parameter.

--help-mode parameter

Syntax	choice of 3 empty leafs **--brief** **--normal** **--full**
Default:	normal
Min Allowed:	0
Max Allowed:	1
Supported by:	netconfd yangcli yangdiff yangdump
Example:	`netconfd --help --help-mode=full`

- **default choice:** normal
- **choice help-mode**
  - **brief**
    - type: empty

- This parameter specifies that brief documentation mode should be used.

- **normal**
  - type: empty
  - This parameter specifies that normal documentation mode should be used.
- **full**
  - type: empty
  - This parameter specifies that full documentation mode should be used.

3.13 --idle-timeout

The **--idle-timeout** parameter controls the maximum number of seconds that the **netconfd** server will wait for a <rpc> PDU, for each session.

If set to 0, then idle state timeouts will be disabled, meaning that no sessions will be deleted while waiting for a <rpc> PDU.

A session will never be considered idle while a notification subscription is active.

It is strongly suggested that this parameter not be disabled, since a denial-of-service attack will be possible if sessions are allowed to remain in the 'idle wait' state forever. A finite number of SSH and NETCONF sessions are supported, so if an attacker simply opened lots of SSH connections to the netconf subsystem, the server would quickly run out of available sessions.

This parameter will affect a <confirmed-commit> operation!

Make sure this timeout interval is larger than the value of the <confirm-timeout> parameter used in the confirmed commit procedure. Otherwise, it is possible that the session will be terminated before the confirm-timeout interval has expired, effectively replacing that timer with this one.

--idle-timeout parameter

Syntax	uint32; 0 == disabled; range 30 .. 360000 seconds (100 hours max)
Default:	3600 (1 hour)
Min Allowed:	0
Max Allowed:	1
Supported by:	netconfd
Example:	`netconfd --idle-timeout=30000`

3.14 --indent

The **--indent** parameter specifies the number of spaces that will be used to add to the indentation level, each time a child node is printed during program operation.

--indent parameter

Syntax	uint32 (0 .. 9)
Default:	3
Min Allowed:	0
Max Allowed:	1
Supported by:	netconfd yangcli yangdiff yangdump
Example:	`netconfd --indent=4`

3.15 --log

The **--log** parameter specifies the file name to be used for logging program messages, instead of STDOUT. It can be used with the optional **--log-append** and **--log-level** parameters to control how the log file is written.

--log parameter

Syntax	string: log file specification
Default:	none
Min Allowed:	0
Max Allowed:	1
Supported by:	netconfd yangcli yangdiff yangdump
Example:	`netconfd --log=~/server.log&`

3.16 --log-append

The **--log-append** parameter specifies that the existing log file (if any) should be appended , instead of deleted. It is ignored unless the **--log** parameter is present.

--log-append parameter

Syntax	empty
Default:	none
Min Allowed:	0
Max Allowed:	1
Supported by:	netconfd yangcli

	yangdiff yangdump
Example:	`netconfd --log-append \` `  --log=~/server.log&`

3.17 --log-level

The **--log-level** parameter controls the verbosity level of messages printed to the log file or STDOUT, if no log file is specified.

The log levels are incremental, meaning that each higher level includes everything from the previous level, plus additional messages.

There are 7 settings that can be used:

- **none**: All logging is suppressed. Use with extreme caution!
- **error**: Error messages are printed, indicating problems that require attention.
- **warn**: Warning messages are printed, indicating problems that may require attention.
- **info**: Informational messages are printed, that indicate program status changes.
- **debug**: Debugging messages are printed that indicate program activity.
- **debug2**: Protocol debugging and trace messages are enabled.
- **debug3**: Very verbose debugging messages are enabled. This has an impact on resources and performance, and should be used with caution.

log-level parameter

Syntax	enumeration: **off** **error** **warn** **info** **debug** **debug2** **debug3**
Default:	--info (--debug for DEBUG builds)
Min Allowed:	0
Max Allowed:	1
Supported by:	netconfd yangcli yangdiff yangdump
Example:	`netconfd --log-level=debug \` `  --log=~/server.log&`

3.18 --max-burst

The **--max-burst** parameter specifies the maximum number of notifications to send in a burst, to one session. Even though TCP will control the transmission rate, this parameter can limit the memory usage due to buffering of notifications waiting to be sent.

The value zero means no limit will be used at all.

--max-burst parameter

Syntax	uint32
Default:	10
Min Allowed:	0
Max Allowed:	1
Supported by:	netconfd yangcli
Example:	`netconfd --max-burst=100`

3.19 --modpath

The **--modpath** parameter specifies the YANG module search path to use while searching for YANG files. It consists of a colon (':') separated list of path specifications, commonly found in Unix, such as the **$PATH** environment variable.

This parameter overrides the **$YUMA_MODPATH** environment variable, if it is present.

--modpath parameter

Syntax	string: list of directory specifications
Default:	**$YUMA_MODPATH** environment variable
Min Allowed:	0
Max Allowed:	1
Supported by:	netconfd yangcli yangdiff yangdump

Example:	`netconfd \`
	`  --modpath=”~/testmodules:~/modules:`
	`~/trunk/netconf/modules”`

3.20 --module

The **--module** parameter is a leaf-list of modules that should be loaded automatically when the program starts.

The program will attempt to load each module in the order the parameters were entered.

For the **netconfd** program, If any modules have fatal errors then the program will terminate.

For the **yangdump** program, each module will be processed as requested.

--module parameter

Syntax	module name or filespec
Default:	none
Min Allowed:	0
Max Allowed:	unlimited
Supported by:	netconfd yangcli yangdump

Example:	`netconfd \`
	`  --module=test1 \`
	`  --module=test2`

3.21 --port

The **--port** parameter specifies the TCP port number(S) that should be used for NETCONF sessions.

This parameter specifies the TCP port numbers to accept NETCONF session from. If any instances of this parameter are found, then the default (port 830) will not be added automatically. Up to 4 port parameter values can be entered.

--port parameter

Syntax	uint32
Default:	830
Min Allowed:	0
Max Allowed:	4
Supported by:	netconfd
Example:	`netconfd --port=22 \` `   --port=830`

3.22 --start

The **--start** parameter is a choice, specifying how the **netconfd** server will load the non-volatile startup configuration at boot-time. If no choice is made at all, then the server will check its data path for the file **startup-cfg.xml**.

- choice **start**
  - **no-startup**
    - type: empty
    - Specifies that no configuration will be loaded at all at boot-time
  - **startup**

- type: string
- Specifies the file name of the boot-time configuration. The server expects this to be an XML configuration file. Unless a path specification is included, the **$YUMA_DATAPATH** environment variable or --datapath parameter will be used to search for the specified file name. No '.xml' extension will be added. The exact file name will be used instead.

start parameter

Syntax	choice: **--no-startup** **--startup**=*filespec*
Default:	first startup-cfg.xml in the data path
Min Allowed:	0
Max Allowed:	1
Supported by:	netconfd
Example:	`netconfd \` `  --startup=$TESTDIR/testrun`

3.23 --startup-error

The **--startup-error** parameter is an enumeration, specifying how the **netconfd** server will treat errors encountered while loading the non-volatile startup configuration at boot-time.

- leaf **startup-error**
  - type: enumeration (stop or continue)
  - Specifies if startup configuration errors will be treated as fatal or recoverable errors.

startup-error parameter

Syntax	enum: stop continue
Default:	continue
Min Allowed:	0
Max Allowed:	1
Supported by:	netconfd
Example:	`netconfd  --startup-error=stop`

3.24 --subdirs

The **--subdirs** parameter controls whether sub-directories should be searched or not, if they are found during a module search.

If false, the file search paths for modules, scripts, and data files will not include sub-directories if they exist in the specified path.

--subdirs parameter

Syntax	boolean
Default:	TRUE
Min Allowed:	0
Max Allowed:	1
Supported by:	yangdump
Example:	`netconfd \` `  --subdirs=false`

3.25 --superuser

The **--superuser** parameter specifies the user name that the **netconfd** server will treat as the super-user account. The 'root' account should be used with caution. It is strongly suggested that root access be disabled in **sshd**, and a different account be used as the NETCONF super-user account.

Any NETCONF session started with this user name will be exempt from access control enforcement. This is especially useful if the current yuma-nacm.yang configuration is preventing access to the <nacm> subtree itself. The super-user account can be used in this situation to repair the mis-configured access control rules.

By default, the name 'superuser' will be accepted as the super-user account, if no value is specified.

To disable all super-user account privileges, set this parameter to a zero-length string.

--superuser parameter

Syntax	*string*
Default:	superuser
Min Allowed:	0
Max Allowed:	1
Supported by:	netconfd
Example:	`netconfd --superuser=admin`

3.26 --target

The **--target** parameter specifies the name of the database that **netconfd** will use as its edit target. There are two targets supported at this time:

- **running**: Edits will be written directly to the <running> configuration. The :startup capability will also be used in this mode. This means that a <copy-config> operation (from <running> to <startup>) must be used to save any edits to non-volatile storage, for use on the next reboot.

- **candidate**: Edits will be written to the <candidate> configuration. The <commit> operation must be used to save any edits in <candidate> configuration into the <running> configuration. The non-volatile storage is updated automatically in this mode, and the <startup> configuration will not be present.

--target parameter

Syntax	enumeration: **running** **candidate**
Default:	candidate
Min Allowed:	0
Max Allowed:	1
Supported by:	netconfd
Example:	`netconfd --target=running`

3.27 --usexmlorder

The **--usexmlorder** parameter specifies that the **netconfd** server should enforce XML ordering when applicable (such as YANG list key leafs entered first). The default is not to check for adherence to strict XML order, as defined by YANG.

--usexmlorder parameter

Syntax	empty
Default:	off
Min Allowed:	0
Max Allowed:	1
Supported by:	netconfd
Example:	`netconfd --usexmlorder`

3.28 --version

The **--version** parameter causes the program version string to be printed, and then the program will exit instead of running as normal.

All Yuma version strings use the same format:

<major>.<minor>.<yang-draft-version>.<svn-build-version>

An example version number that may be printed:

```
netconfd 0.9.6.390
```

This indicates that the **yangdump** program version is '0.9', it supports YANG draft version '-06', and the subversion build identifier is '390'.

This parameter can be combined with the **--help** parameter.

--version parameter

Syntax	empty
Default:	none
Min Allowed:	0
Max Allowed:	1
Supported by:	netconfd yangcli yangdiff yangdump
Example:	`netconfd --version`

3.29 --warn-idlen

The **--warn-idlen** parameter controls whether identifier length warnings will be generated.

The value zero disables all identifier length checking. If non-zero, then a warning will be generated if an identifier is defined which has a length is greater than this amount.

--warn-idlen parameter

Syntax	uint32: 0 to disable, or 8 .. 1023
Default:	64
Min Allowed:	0
Max Allowed:	1
Supported by:	netconfd yangcli yangdiff yangdump
Example:	`netconfd --warn-idlen=50`

3.30 --warn-linelen

The **--warn-linelen** parameter controls whether line length warnings will be generated.

The value zero disables all line length checking. If non-zero, then a warning will be generated if a YANG file line is entered which has a length is greater than this amount.

Tab characters are counted as 8 spaces.

--warn-linelen parameter

Syntax	uint32: 0 to disable, or 40 .. 4095
Default:	72
Min Allowed:	0
Max Allowed:	1
Supported by:	netconfd yangcli yangdiff yangdump
Example:	`netconfd --warn-linelen=79`

3.31 --warn-off

The **--warn-off** parameter suppresses a specific warning number.

The error and warning numbers, and the default messages, can be viewed with the yangdump program by using the **--show-errors** configuration parameter.

The specific warning message will be disabled for all modules. No message will be printed and the warning will not count towards the total for that module.

--warn-off parameter

Syntax	uint32: 400 .. 899
Default:	none
Min Allowed:	0
Max Allowed:	499
Supported by:	netconfd yangcli yangdiff yangdump
Example:	`netconfd --warn-off=435` `# revision order not descending`

3.32 --with-startup

The **--with-startup** parameter controls whether the server will support the :startup capability or not. This is a memory intensive capability, and setting this parameter to 'false' will reduce memory usage during normal operations. The non-volatile copy of the <running> configuration will not be saved automatically if this capability is enabled. Instead, a <copy-config>operation from the <running> to <startup> configuration will be needed.

--with-startup parameter

Syntax	boolean
Default:	FALSE
Min Allowed:	0
Max Allowed:	1
Supported by:	netconfd
Example:	`netconfd --with-startup=true`

3.33 --with-url

The **--with-url** parameter controls whether the server will support the :url capability or not. This capability requires file system storage.

--with-url parameter

Syntax	boolean
Default:	TRUE
Min Allowed:	0
Max Allowed:	1
Supported by:	netconfd
Example:	`netconfd --with-url=false`

3.34 --with-validate

The **--with-validate** parameter controls whether the server will support the :validate capability or not. This is a memory intensive capability, and setting this parameter to 'false' will reduce memory usage during <edit-config> operations.

--with-validate parameter

Syntax	boolean
Default:	TRUE
Min Allowed:	0
Max Allowed:	1
Supported by:	netconfd
Example:	`netconfd --with-validate=false`

3.35 --yuma-home

The **--yuma-home** parameter specifies the project directory root to use when searching for files.

If present, this directory location will override the '**$YUMA_HOME** environment variable, if it is set. If this parameter is set to a zero-length string, then the $**YUMA_HOME** environment variable will be ignored.

The following directories are searched when either the **$YUMA_HOME** environment variable or this parameter is set:

- **$YUMA_HOME/modules**
  - This sub-tree is searched for YANG files.
- **$YUMA_HOME/data**
  - This directory is searched for data files.
- **$YUMA_HOME/scripts**
  - This directory is searched for **yangcli** script files.

yuma-home parameter

Syntax	string: directory specification
Default:	**$YUMA_HOME** environment variable
Min Allowed:	0
Max Allowed:	1
Supported by:	netconfd yangcli yangdiff yangdump

Example:	`netconfd \`
	`  --yuma-home=~/sw/netconf \`
	`  --log=~/server.log&`