

# **Yuma Tools® User Manual**

---

YANG-Based Unified Modular Automation Tools

## Table Of Contents

### Yuma Tools User Manual

1 Preface.....	8
1.1 Legal Statements.....	8
1.2 Restricted Rights Legend.....	8
1.3 Additional Resources.....	8
1.3.1 WEB Sites.....	8
1.3.2 Mailing Lists.....	9
1.4 Conventions Used in this Document.....	9
2 Summary.....	10
2.1 What is Yuma?.....	10
2.2 Intended Audience.....	11
3 Introduction.....	12
3.1 System Components.....	12
3.1.1 YANG.....	14
3.1.2 NETCONF.....	16
3.1.3 YANG-based Automation.....	19
3.1.4 YANG Language Extensions.....	24
3.1.5 YANG Compiler.....	25
3.1.6 YANG Module Library.....	25
3.1.7 YANG Files.....	28
3.1.8 NETCONF Managers.....	28
3.1.9 NETCONF Agents.....	28
4 System Configuration.....	30
4.1 Environment Variables.....	30
4.1.1 \$HOME.....	31
4.1.2 \$YUMA_HOME.....	31
4.1.3 \$YUMA_INSTALL.....	32
4.1.4 \$YUMA_MODPATH.....	32
4.1.5 \$YUMA_DATAPATH.....	33
4.1.6 \$YUMA_RUNPATH.....	33
4.2 Searching for YANG Files.....	34
4.2.1 Parameter Searches.....	35
4.2.2 Import/Include Searches.....	36
4.2.3 File Search Paths.....	37
4.3 Configuration Files.....	39
4.3.1 XML Configuration Files.....	40
4.3.2 Text Configuration Files.....	41
4.4 Bootstrap CLI.....	43
4.5 Configuration Parameters.....	43
4.5.1 Parameter Syntax.....	43
4.5.2 ncx:cli Extension.....	44
4.5.3 ncx:default-parm Extension.....	45
5 yangdump User Guide.....	46
5.1 Introduction.....	46
5.1.1 Features.....	46
5.1.2 Starting yangdump.....	49

# Yuma Tools User Manual

5.1.3 Stopping yangdump.....	50
5.1.4 Configuration Parameter List.....	50
5.2 Validating YANG Files.....	52
5.2.1 Yangdump Validation Messages.....	52
5.2.2 Validation Example.....	53
5.3 Translating YANG to Other Formats.....	54
5.3.1 YIN Format.....	54
5.3.2 HTML Translation.....	56
5.3.3 XSD Translation.....	59
5.3.4 SQL Translation.....	62
5.3.5 SQL Documentation Database Translation.....	63
5.3.6 Canonical YANG Translation.....	63
5.3.7 Copy and Rename YANG Files.....	63
5.3.8 Agent Instrumentation H File Generation.....	64
5.3.9 Agent Instrumentation C File Generation.....	64
6 yangdiff User Guide.....	65
6.1 Introduction.....	65
6.1.1 Features.....	65
6.1.2 Starting yangdiff.....	66
6.1.3 Stopping yangdiff.....	66
6.1.4 Configuration Parameter List.....	67
6.2 Comparing YANG Modules.....	67
6.3 Diff Reports.....	68
6.3.1 Terse Report.....	68
6.3.2 Normal Report.....	69
6.3.3 Revision Statement.....	69
7 yangcli User Guide.....	71
7.1 Introduction.....	71
7.1.1 Features.....	71
7.1.2 Starting yangcli.....	73
7.1.3 Stopping yangcli.....	74
7.1.4 Statements.....	74
7.1.5 Commands.....	75
7.1.6 Variables.....	77
7.1.7 Files.....	79
7.1.8 Scripts.....	80
7.1.9 Configuration Parameter List.....	81
7.2 Invoking Commands.....	82
7.2.1 Command Prompt.....	82
7.2.2 Command Name.....	85
7.2.3 ncx:default-parm Extension.....	85
7.2.4 Parameter Mode Escape Commands.....	86
7.2.5 Using XPath Expressions.....	87
7.2.6 Special Parameter Handling.....	88
7.2.7 Command Completion.....	89
7.2.8 Command Line Editing.....	90
7.2.9 Command History.....	91
7.2.10 Command Responses.....	92
7.3 NETCONF Sessions.....	93
7.3.1 Connection Startup Screen.....	93
7.3.2 Server Tailored Context.....	94

## Yuma Tools User Manual

7.3.3 Retrieving Data.....	96
7.3.4 Modifying Data.....	96
7.3.5 Using Notifications.....	98
7.3.6 Configuration Parameters That Affect Sessions.....	99
7.3.7 Trouble-shooting NETCONF Session Problems.....	99
7.4 Command Reference.....	101
7.4.1 cd.....	101
7.4.2 close-session.....	102
7.4.3 commit.....	103
7.4.4 connect.....	104
7.4.5 copy-config.....	106
7.4.6 create.....	108
7.4.7 create-subscription.....	111
7.4.8 delete.....	113
7.4.9 delete-config.....	114
7.4.10 discard-changes.....	116
7.4.11 edit-config.....	117
7.4.12 eventlog.....	119
7.4.13 fill.....	121
7.4.14 get.....	123
7.4.15 get-config.....	125
7.4.16 get-locks.....	127
7.4.17 get-my-session.....	129
7.4.18 get-schema.....	130
7.4.19 help.....	132
7.4.20 history.....	136
7.4.21 insert.....	138
7.4.22 kill-session.....	141
7.4.23 list.....	142
7.4.24 load.....	146
7.4.25 lock.....	147
7.4.26 merge.....	149
7.4.27 mgrload.....	151
7.4.28 no-op.....	152
7.4.29 pwd.....	153
7.4.30 quit.....	153
7.4.31 recall.....	154
7.4.32 release-locks.....	155
7.4.33 replace.....	155
7.4.34 restart.....	158
7.4.35 run.....	158
7.4.36 save.....	160
7.4.37 set-log-level.....	161
7.4.38 set-my-session.....	161
7.4.39 sget.....	163
7.4.40 sget-config.....	165
7.4.41 show.....	169
7.4.42 shutdown.....	173
7.4.43 unlock.....	174
7.4.44 validate.....	175
7.4.45 xget.....	176

# Yuma Tools User Manual

7.4.46 xget-config.....	179
<b>8 netconfd User Guide.....</b>	<b>183</b>
<b>8.1 Introduction.....</b>	<b>183</b>
8.1.1 Features.....	183
8.1.2 Setting the Server Profile.....	185
8.1.3 Loading YANG Modules.....	185
8.1.4 Starting netconfd.....	185
8.1.5 Stopping netconfd.....	186
8.1.6 Signal Handling.....	187
8.1.7 Error Handling.....	187
8.1.8 Module Summary.....	188
8.1.9 Notification Summary.....	189
8.1.10 Operation Summary.....	189
8.1.11 Configuration Parameter List.....	190
<b>8.2 Capabilities.....</b>	<b>191</b>
8.2.1 :candidate.....	191
8.2.2 :confirmed-commit.....	192
8.2.3 :interleave.....	192
8.2.4 :netconf-monitoring.....	192
8.2.5 :notification.....	193
8.2.6 :rollback-on-error.....	193
8.2.7 :schema-retrieval.....	193
8.2.8 :startup.....	193
8.2.9 :validate.....	194
8.2.10 :with-defaults.....	194
8.2.11 :writable-running.....	195
8.2.12 :xpath.....	195
<b>8.3 Databases.....</b>	<b>195</b>
8.3.1 Database Locking.....	196
8.3.2 Using the <candidate> Database.....	197
8.3.3 Using the <running> Database.....	197
8.3.4 Using the <startup> Database.....	197
<b>8.4 Sessions.....</b>	<b>197</b>
8.4.1 User Names.....	198
8.4.2 Session ID.....	198
8.4.3 Server <hello> Message.....	198
8.4.4 Client <hello> Message.....	201
8.4.5 RPC Request Processing.....	202
8.4.6 Session Termination.....	202
<b>8.5 Error Reporting.....</b>	<b>203</b>
8.5.1 <error-severity> Element.....	203
8.5.2 <error-tag> Element.....	203
8.5.3 <error-app-tag> Element.....	204
8.5.4 <error-path> Element.....	205
8.5.5 <error-message> Element.....	206
8.5.6 <error-info> Element.....	206
8.5.7 instance-required Error Example.....	206
8.5.8 missing-choice Error Example.....	208
8.5.9 no-matches Error Example.....	209
8.5.10not-in-range Error Example.....	210
<b>8.6 Protocol Operations.....</b>	<b>211</b>

# Yuma Tools User Manual

8.6.1 <close-session>.....	212
8.6.2 <commit>.....	213
8.6.3 <copy-config>.....	214
8.6.4 <create-subscription>.....	216
8.6.5 <delete-config>.....	218
8.6.6 <discard-changes>.....	220
8.6.7 <edit-config>.....	221
8.6.8 <get>.....	223
8.6.9 <get-config>.....	225
8.6.10 <get-my-session>.....	228
8.6.11 <get-schema>.....	229
8.6.12 <kill-session>.....	231
8.6.13 <load>.....	232
8.6.14 <lock>.....	233
8.6.15 <no-op>.....	235
8.6.16 <restart>.....	236
8.6.17 <set-log-level>.....	236
8.6.18 <set-my-session>.....	238
8.6.19 <shutdown>.....	239
8.6.20 <unlock>.....	240
8.6.21 <validate>.....	241
8.7 Access Control.....	243
8.7.1 NACM Module Structure.....	244
8.7.2 Users and Groups.....	245
8.7.3 Creating New Groups.....	246
8.7.4 Access Control Modes.....	246
8.7.5 Permissions.....	247
8.7.6 Default Enforcement Behavior.....	247
8.7.7 Access Control Algorithm.....	247
8.7.8 Module Access Control Rules.....	250
8.7.9 RPC Access Control Rules.....	251
8.7.10 Data Access Control Rules.....	252
8.8 Monitoring.....	253
8.8.1 Using Subtree Filters.....	254
8.8.2 Using XPath Filters.....	257
8.8.3 System Information.....	260
8.8.4 Interfaces Information.....	260
8.8.5 NETCONF State Information.....	260
8.9 Notifications.....	260
8.9.1 Subscriptions.....	260
8.9.2 Notification Log.....	261
8.9.3 Using Notification Filters.....	261
8.9.4 <notification> Element.....	261
8.9.5 <replayComplete> Event.....	262
8.9.6 <notificationComplete> Event.....	262
8.9.7 <sysStartup> Event.....	263
8.9.8 <sysSessionStart> Event.....	264
8.9.9 <sysSessionEnd> Event.....	265
8.9.10 <sysConfigChange> Event.....	266
8.9.11 <sysCapabilityChange> Event.....	268
8.9.12 <sysConfirmedCommit> Event.....	270

## Yuma Tools User Manual

9 CLI Reference.....	272
9.1 --access-control.....	272
9.2 --autocomp.....	272
9.3 --autohistory.....	273
9.4 --autoload.....	273
9.5 --bad-data.....	274
9.6 --batch-mode.....	274
9.7 --config.....	275
9.8 --datapath.....	275
9.9 --default-module.....	276
9.10 --default-style.....	276
9.11 --defnames.....	277
9.12 --dependencies.....	277
9.13 --deviation.....	278
9.14 --difftype.....	279
9.15 --display-mode.....	280
9.16 --eventlog-size.....	281
9.17 --exports.....	281
9.18 --format.....	282
9.19 --fixorder.....	283
9.20 --header.....	284
9.21 --help.....	284
9.22 --help-mode.....	284
9.23 --hello-timeout.....	285
9.24 --html-div.....	286
9.25 --html-toc.....	287
9.26 --identifiers.....	287
9.27 --idle-timeout.....	288
9.28 --indent.....	289
9.29 --log.....	289
9.30 --log-append.....	290
9.31 --log-level.....	290
9.32 --max-burst.....	291
9.33 --modpath.....	291
9.34 --module.....	292
9.35 --modversion.....	292
9.36 --new.....	293
9.37 --objview.....	294
9.38 --old.....	294
9.39 --output.....	295
9.40 --password.....	296
9.41 --port.....	297
9.42 --runpath.....	297
9.43 --run-command.....	298
9.44 --run-script.....	298
9.45 --server.....	298
9.46 --show-errors.....	299

## Yuma Tools User Manual

9.47 --simurls.....	300
9.48 --start.....	300
9.49 --subdirs.....	301
9.50 --subtree.....	302
9.51 --superuser.....	302
9.52 --target.....	303
9.53 --timeout.....	303
9.54 --unified.....	304
9.55 --urlstart.....	305
9.56 --user.....	305
9.57 --usexmlorder.....	306
9.58 --version.....	306
9.59 --versionnames.....	307
9.60 --warn-idlen.....	307
9.61 --warn-linelen.....	308
9.62 --warn-off.....	308
9.63 --with-startup.....	309
9.64 --with-validate.....	309
9.65 --xsd-schemaloc.....	309
9.66 --yuma-home.....	310
10 Error Reference.....	311
10.1 Error Messages.....	311

# 1 Preface

## 1.1 Legal Statements

Copyright 2009 Netconf Central, Inc., All Rights Reserved.

## 1.2 Restricted Rights Legend

This software is provided with RESTRICTED RIGHTS.

Use, duplication, or disclosure by the U.S. Government is subject to restrictions set forth in subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227-7013 or subparagraphs(c)(1) and (2) of the Commercial Computer Software - Restricted Rights at 48 CFR 52.227-19, as applicable.

The "Manufacturer" for purposes of these regulations is Netconf Central, Inc., 374 Laguna Terrace, Simi Valley, California 93065 U.S.A.

## 1.3 Additional Resources

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

Yuma® Installation and Quickstart Guide

Depending on the version of Yuma you purchased, other documentation includes:

Yuma® Developer Manual

To obtain additional support you may email Netconf Central at the e-mail address  
[support@netconfcentral.com](mailto:support@netconfcentral.com)

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.

### 1.3.1 WEB SITES

- **Netconf Central**
  - <http://www.netconfcentral.org>
  - Free information on NETCONF and YANG, tutorials, on-line YANG module validation and documentation database
- **Yang Central**
  - <http://www.yang-central.org>
  - Free information and tutorials on YANG, free YANG tools for download
- **NETCONF Working Group Wiki Page**

## Yuma Tools User Manual

- <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

### 1.3.2 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.

## 1.4 Conventions Used in this Document

The following formatting conventions are used throughout this document:

### Documentation Conventions

Convention	Description
<b>--foo</b>	CLI parameter foo
<b>&lt;foo&gt;</b>	XML parameter foo
<b>foo</b>	<b>yangcli</b> command or parameter
<b>\$\$FOO</b>	Environment variable FOO
<b>\$\$foo</b>	<b>yangcli</b> global variable foo
some text	Example command or PDU
some text	Plain text

## 2 Summary

### 2.1 What is Yuma?

Yuma is a set of programs providing a complete network management system and development environment, which implements the following standards:

- Network Configuration Protocol (RFC 4741)
- NETCONF over SSH (RFC 4742)
- NETCONF Notifications (RFC 5277)
- SSH2 (RFC 4252 - 4254)
- XML 1.0
- XPath 1.0
- YANG Data modeling language (RFC xxxx)

The following programs are included in the Yuma suite:

- **yangdump**: validates YANG modules and uses them to generate other formats, such as HTML, XSD, SQL, and C source code
- **yangdiff**: reports semantic differences between two revisions of a YANG module, and generates YANG revision statements
- **yangcli**: NETCONF over SSH client, providing a simple but powerful command line interface for management of any NETCONF content defined in YANG
- **netconfd**: NETCONF over SSH server, providing complete and automated support for the YANG content accessible with the NETCONF protocol
- **netconf-subsystem**: thin client used to allow OpenSSH to communicate with the netconfd program. This is documented as part of the **netconfd** program, since they must be used together.

Although any arbitrary YANG file can be automatically supported by Yuma, the following content (YANG modules) is built into the **netconfd** server, and supported by the **yangcli** client:

- **yuma-netconf.yang**: all the NETCONF protocol operations, including all YANG extensions to the NETCONF protocol (RFC 4741). This file contains meta-data used in the yangcli and netconfd programs, which is not available in the ietf-netconf.yang version.
- **ietf-netconf-state.yang**: the standard NETCONF monitoring module in progress by the NETCONF WG (draft-ietf-netconf-monitoring-07.txt)
- **ietf-with-defaults.yang**: the standard NETCONF default value control module in progress by the NETCONF WG (draft-ietf-netconf-with-defaults-03.txt)
- **yuma-interfaces.yang**: interfaces monitoring and configuration scaffolding.
- **yuma-mysession.yang**: NETCONF session customization operations
- **notifications.yang**: the standard NETCONF create-subscription command to start receiving NETCONF notifications (RFC 5277)

- **nc-notifications.yang**: the standard NETCONF notifications (RFC 5277)
- **yuma-proc.yang**: /proc file system monitoring information
- **yuma-system.yang**: Proprietary system group and common notifications
- **yuma-nacm.yang**: Proprietary NETCONF Access Control Model
- **test/\*.yang**: Several modules are included for testing YANG and NETCONF behavior.

## 2.2 Intended Audience

This document is intended for users of the programs in the Yuma suite.

It contains the following information:

- Introduction to YANG and NETCONF based Network Management
- Yuma Configuration
- Yuma User Guides
- Yuma CLI Reference
- Yuma Error Reference

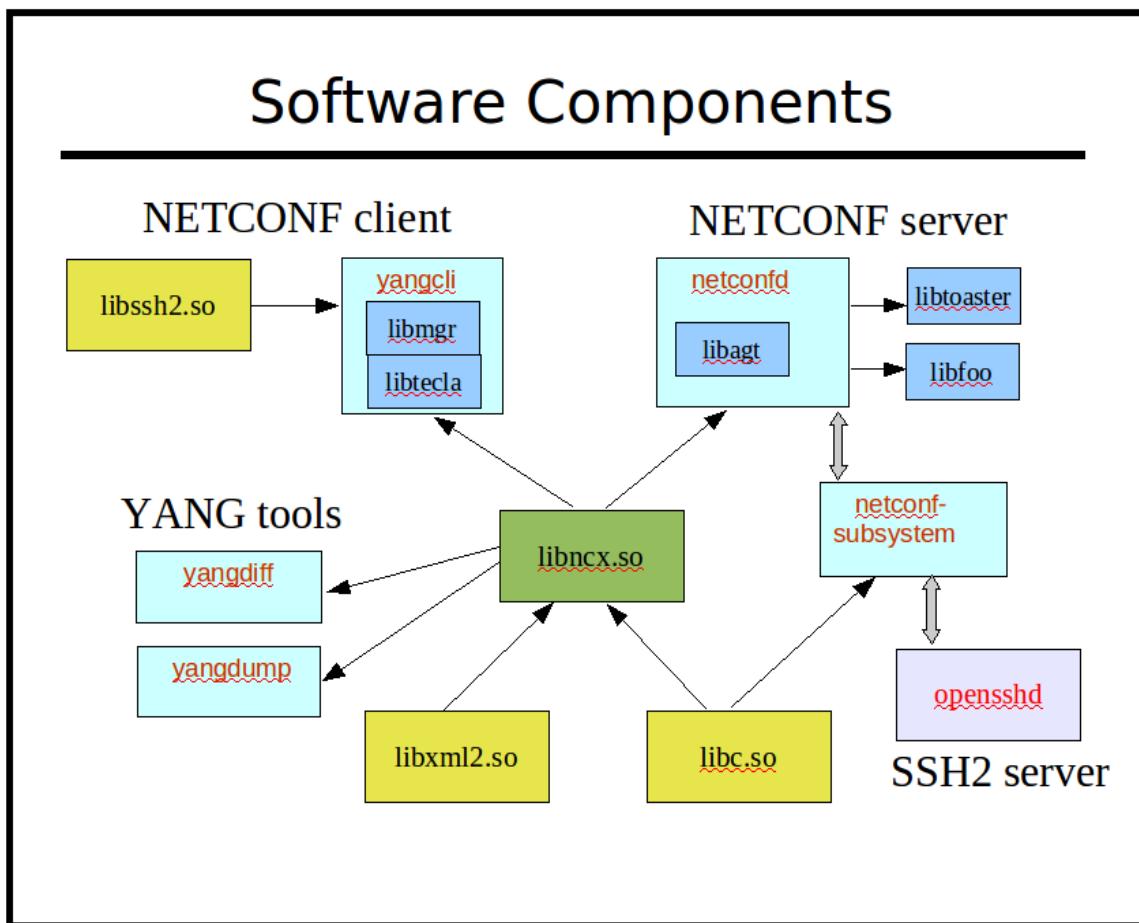
## 3 Introduction

The Yuma suite provides automated support for development and usage of network management information.

All management data is defined with the YANG data modeling language.

All management operations are encoded in XML 1.0 and performed with standard NETCONF protocol operations.

### 3.1 System Components



The following external program is used by Yuma, and needs to be pre-installed:

- **opensshd**
  - The SSH2 server code does not link with Yuma. Instead, the **netconf-subsystem** program is invoked, and local connections are made to the **netconfd** server from this SSH2 subsystem.

## Yuma Tools User Manual

The following external libraries are used by Yuma, and need to be pre-installed:

- **libc (or glibc)**
  - unix system library
- **libssh2**
  - SSH2 client library
- **libxml2**
  - xmlTextReader XML parser
  - pattern support

The following external library is built within Yuma and does not need to be pre-installed:

- **libtecla**
  - command line support for **yangcli**

The following shared library is built by Yuma and used by almost all of its programs:

- **libncx**
  - YANG parser
  - YANG validation
  - basic NETCONF support
  - XPath support
  - configuration database support

The following libraries are built by Yuma, and used within executables:

- **libagt**
  - NETCONF server support
- **libmgr**
  - NETCONF client support

The following binaries are built by Yuma:

- **netconfd**
  - NETCONF server
- **netconf-subsystem**
  - thin client between opensshd and NETCONF server
- **yangcli**
  - NETCONF client
- **yangdump**
  - YANG validation
- **yangdiff**
  - YANG compare

The following sample netconfd module instrumentation library is provided as an example. These libraries (e.g., libfoo.so) can only be created with the Yuma SDK. Refer to the Yuma Developer's Guide for details on creating server instrumentation libraries.

- **libtoaster**

- Server instrumentation code for the YANG module libtoaster.yang.

### 3.1.1 YANG

A YANG module define the semantics and syntax of a specific management feature. They are similar to SMIv2 (MIB) modules, but much more powerful and extensible. YANG provides the ability to define a detailed programmatic interface utilizing all protocol features:

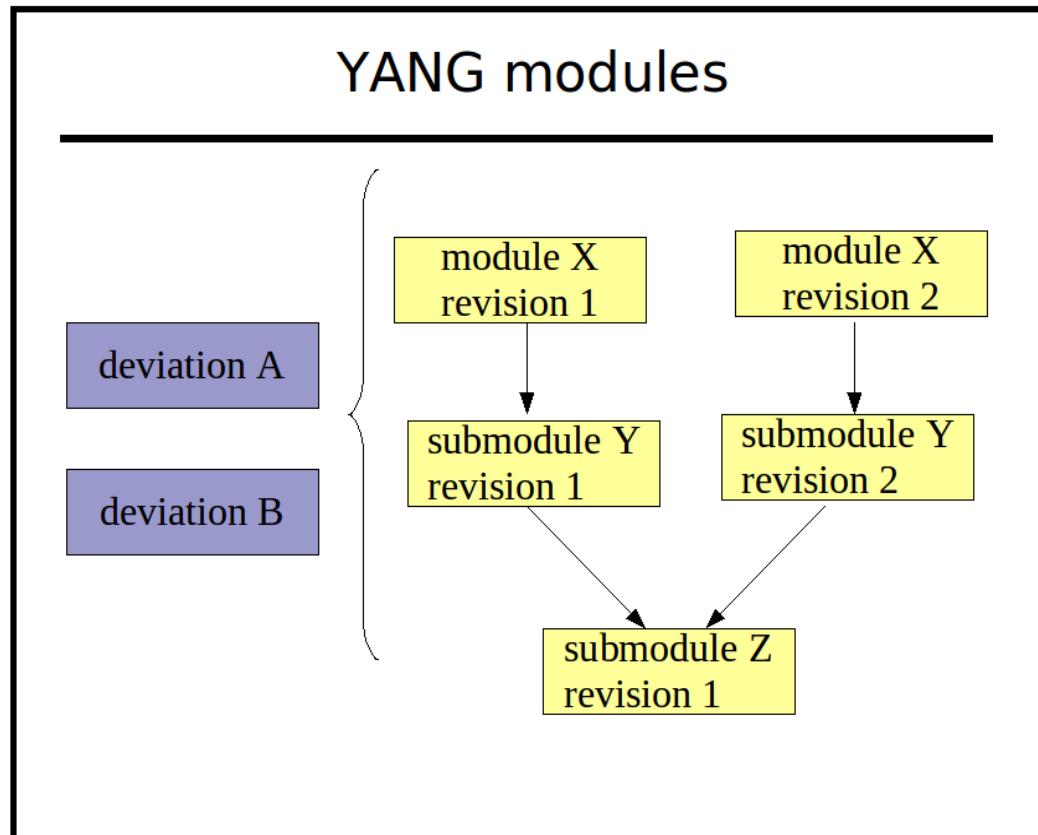
- reusable derived data types
- reusable groupings of objects
- RPC operations
- database objects
- notifications

Network management software developers creating a new management feature start by defining the YANG module(s) for the NETCONF representation of the feature. This can include any mixture of new operations, data, and notifications. Existing YANG modules can be augmented as well.

YANG provides complex nested data structures and choices, which allows data modelers to design management interfaces which closely resemble the native data structures within the server implementation.

It is easy to get started with YANG, and there are many optional advanced features that can be utilized as well. YANG provides many machine-readable constructs which allow Yuma to automate many aspects of network management software development.

Semant



## Yuma Tools User Manual

A YANG module can be a single file, or it can be split into an arbitrary number of files, using submodules. A YANG submodule is essentially the same as a main module, except that the namespace URI value is shared between the main module and all its submodules.

A submodule is referenced with the include statement instead of the import statement.

Submodules can also include other submodules, except a loop may not be formed by the include statements.

Conceptually, the module is not nested. All definitions in submodules appear at the top level of the YANG module, even submodules included by other submodules.

All YANG modules and submodules have revision dates. The example shows a simple version number, but the actual revision strings are date strings in the form 'YYYY-MM-DD'.

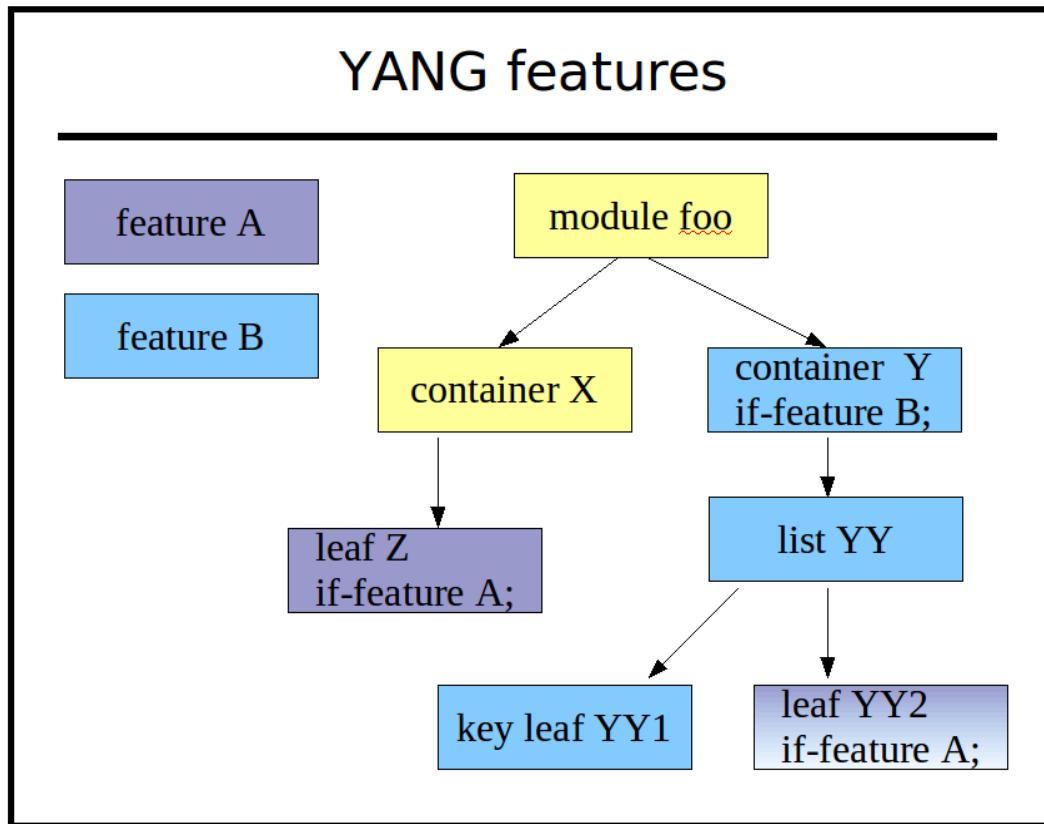
Yuma programs support concurrent usage of different revisions of the same module or submodule. This can occur via groupings from external modules within the YANG language. Only one revision of a module can be imported into a single module or submodule, but any of these files may in turn import other modules. It is possible that a different version of the same module could be indirectly imported in this case.

Deviation modules are normal YANG modules, except they only contain deviation statements. These deviation statements are used to alter (patch) the YANG modules with implementation-specific differences.

A deviation module can contain any number of deviation statements, and they can apply to an arbitrary number of objects, from any module. Multiple deviation statements for the same target will be combined by the server before using them, and all deviate statements for the same object will be validated together, as if they were all contained in the same deviation statement. The order of the deviation statements is irrelevant.

Deviations modules are processed first, and the deviation statements save for later. The import statements are ignored, unlike real module processing.

Since deviation modules are not identified in any way, Yuma programs use the **--module** parameter to refer to a normal YANG module or submodule, and the **--deviation** parameter to refer to a deviation module.



The YANG feature statement is used to define a conceptual partition within the module.

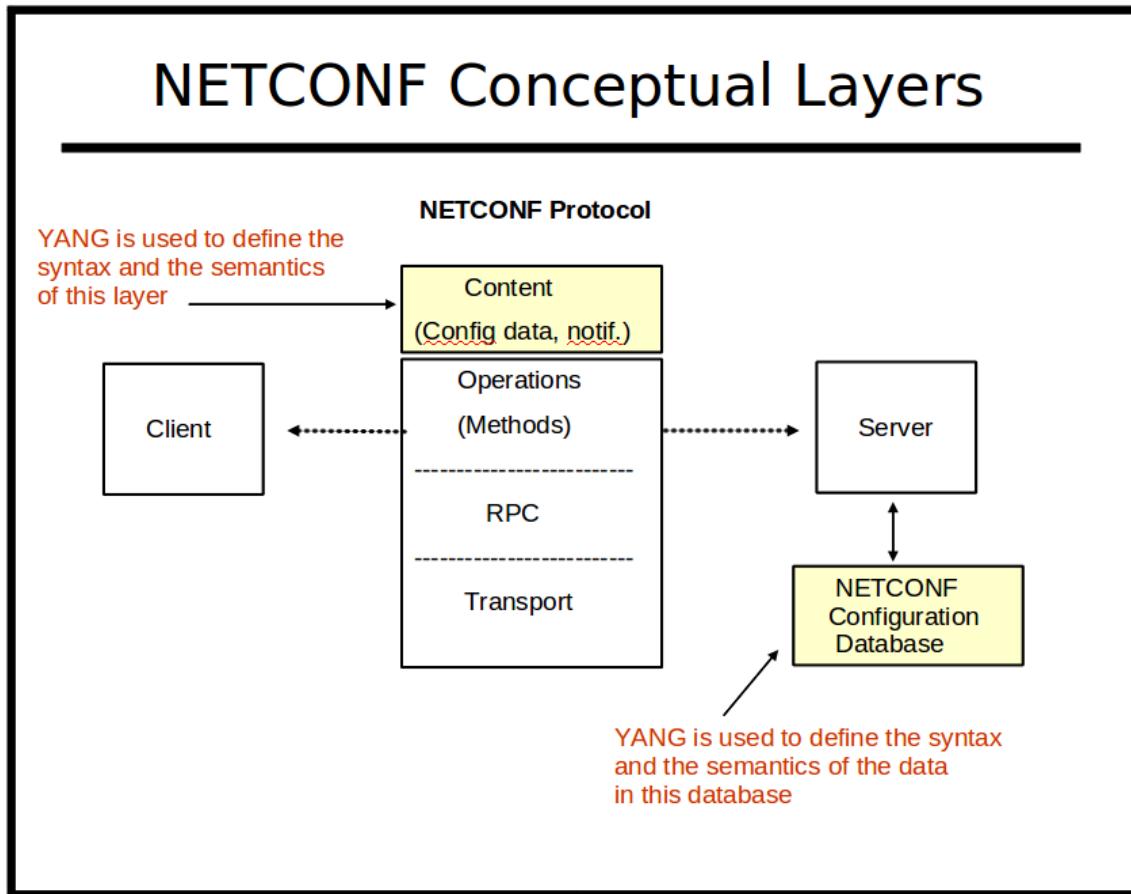
Objects that contain the if-feature statement for the corresponding feature are part of the feature.

If the server does not advertise a feature in its <capabilities>, then it is not supported, and all the objects that are part of the feature are not supported.

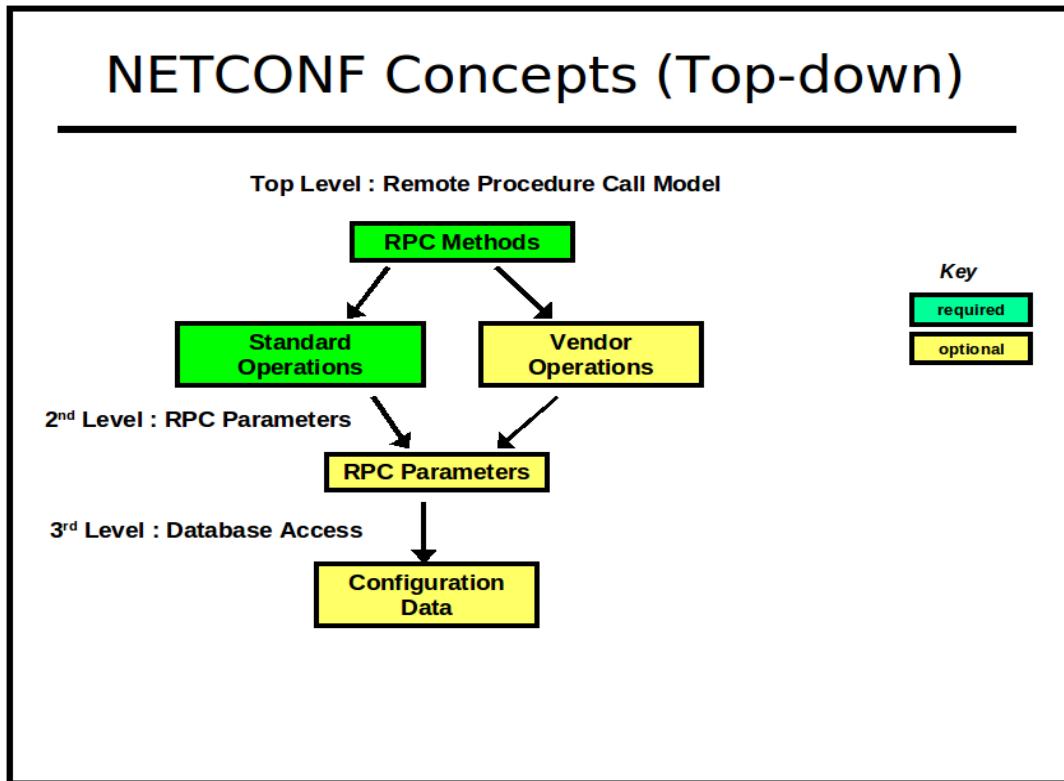
Multiple if-feature statements form a logical AND expression. All the referenced features must be enabled for the object to be available. In the example above, leaf 'YY2' is not present unless feature A and B are both advertised by the server.

### 3.1.2 NETCONF

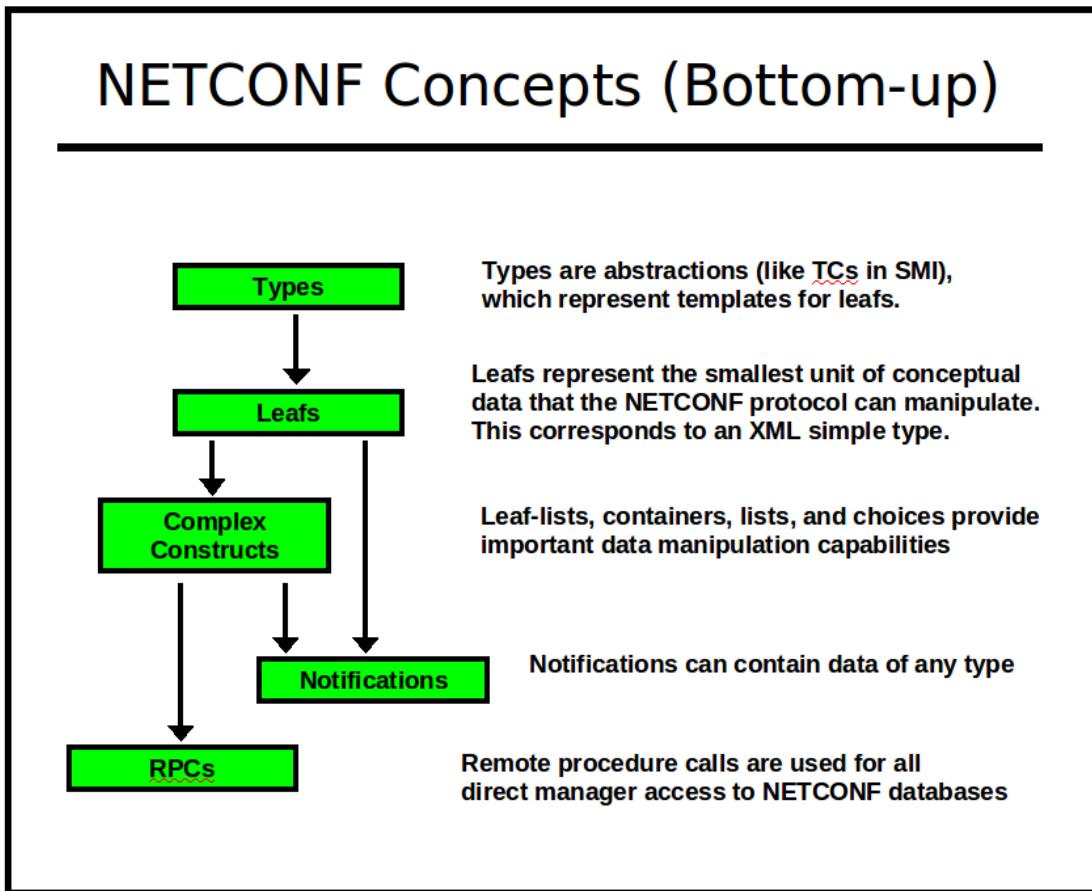
The mandatory components of the NETCONF protocol are defined in RFC 4741 and RFC 4742.



The NETCONF protocol is used to provide secure access all YANG content. The server maintains a database which is accessed as if it was an XML instance document.



Data can be retrieved with XML (subtree) or XPath filters. Changes can be validated before being activated. Databases can be locked to prevent multiple managers from interfering with each other. Custom operations can be used to perform complex actions and perhaps return some data as well.



NETCONF can utilize several secure transport protocols. The mandatory transport (SSH2) is used by Yuma. The **OpenSSH** server is used in the **netconfd** implementation, and **libssh2** library is used in the **yangcli** implementation, to provide all SSH2 layer support.

By default, TCP port 830 (netconf-over-ssh) is used for all NETCONF communications between **yangcli** and **netconfd**. TCP port 22 (ssh) is also supported by default, and additional TCP ports can be configured.

NETCONF security is session-based. Privileges are granted to a session based on the username provided in the SSH connection setup.

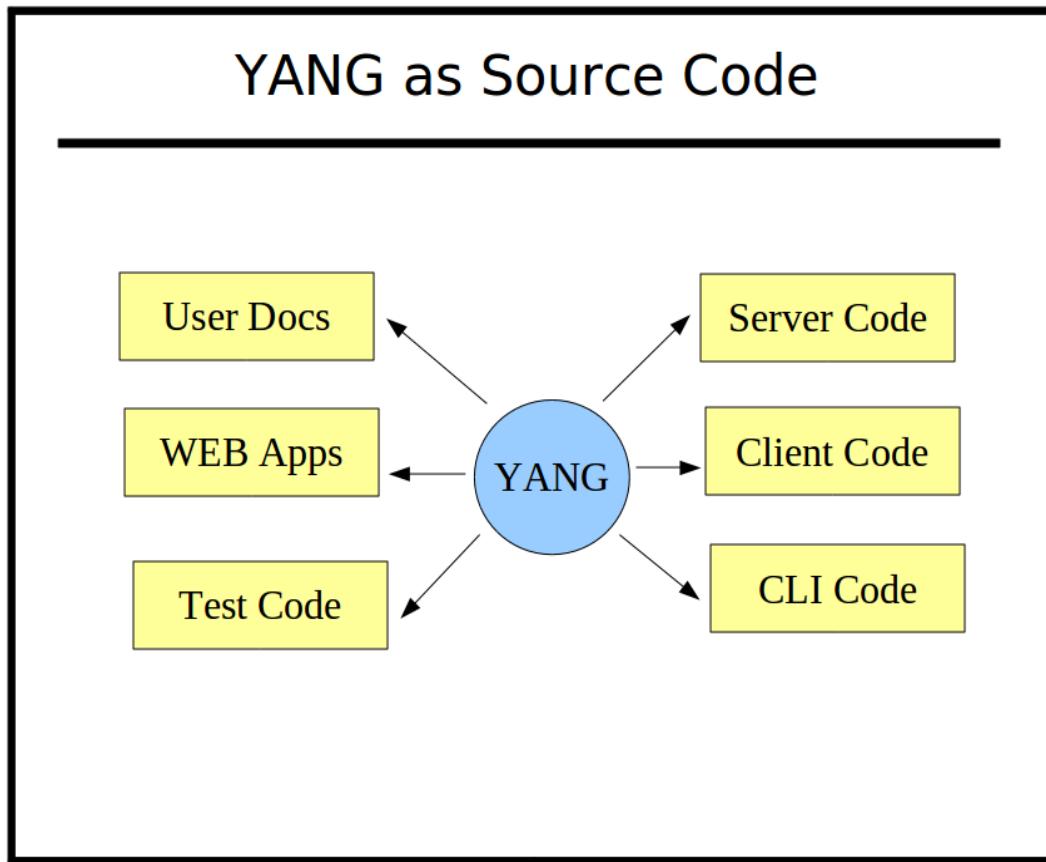
Access control is configurable (via **yuma-nacm.yang**), based on group membership. The access control rules permit or deny access to one or more groups, to a subset of the YANG content. Separate defaults for read, write, and exec (RPC operation) access are provided.

### 3.1.3 YANG-BASED AUTOMATION

Yuma is a 100% “native YANG” implementation. This means that YANG modules are used directly by all the tools to control all aspects of NETCONF protocol usage. There are no lossy translations, or complicated configuration steps, in order to use a YANG module. Simply load a module and start using it.

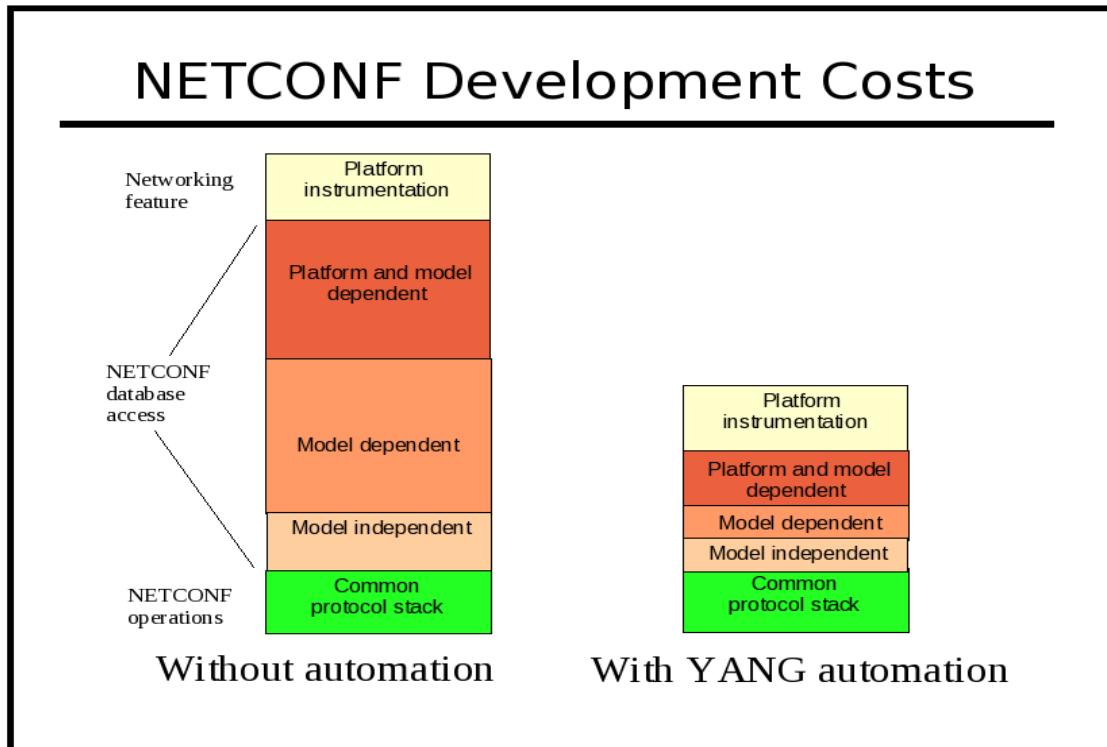
The automation concepts will be familiar to SNMP developers who use SMIv2 to write MIB modules. The SMIv2 language contains enough machine-readable clauses so a client and server can automate certain aspects of the SNMP protocol implementation.

YANG does the same thing for NETCONF developers, only 10 times better.



There are many more machine-readable constructs in YANG, and more powerful data modeling features. The complicated YANG features are optional, so traditional 'DESCRIPTION clause' based semantics are still supported.

The more machine-readable YANG clauses that are used, the more the **yangcli** client and **netconfd** server can automate the entire NETCONF protocol implementation.



The YANG language includes many ways to specify conditions for database validity, which traditionally are only documented in DESCRIPTION clauses:

### YANG Automation Constructs

<b>YANG statement</b>	<b>description</b>
<b>config boolean;</b>	The <b>config</b> statement indicates if the object is writable, or read-only. The server uses this information when automatically skipping config=false entries for the <get-config> operation.
<b>default string;</b>	The <b>default</b> statement specifies the mandatory-to-use default value, if no leaf is provided. Unlike SMIv2 DEFVAL, it is not a suggestion, and the client can rely on it. Defaults can be specified in <b>typedef</b> or <b>leaf</b> statements. If both are defined, then the leaf default will be used.
<b>deviation deviation-target-path { ... }</b>	The <b>deviation</b> statement allows any YANG object be customized for a particular platform or implementation.. The tools can automatically support the altered objects, based on the sub-statements within the <b>deviation</b> statement. These changes can be of any nature, even those normally not allowed in YANG. The intent of the deviation

## Yuma Tools User Manual

	statement os to accurately describe the object implementation, so the tools can automate the protocol operations correctly, even for non-standard implementations.
<b>error-app-tag</b> <i>apptag-string</i> ;	The <b>error-app-tag</b> statement can be used within the <b>range</b> , <b>length</b> , and <b>pattern</b> statements. If a value is invalid due to the corresponding error, then the <error-app-tag> field in the <rpc-error> sent by the server will be set to the 'apptag-string' value.
<b>error-message</b> <i>errmsg-string</i> ;	The <b>error-message</b> statement can be used within the <b>range</b> , <b>length</b> , and <b>pattern</b> statements. If a value is invalid due to the corresponding error, then the <error-message> field in the <rpc-error> sent by the server will be set to the 'errmsg-string' value.
<b>extension</b>	The <b>extension</b> statement allows a vendor to add language extensions, and all YANG implementations must be able to parse the extension correctly. However, only implementations which actually understand the extension will support it. All others will simply ignore the extension.
<b>feature</b>	The <b>feature</b> statement allows a module to be conceptually partitioned into mandatory and conditional object groups. All objects with the corresponding if-feature statement will be present only if the feature is supported by the server.
<b>if-feature</b> <i>feature-name</i> ;	Construct containing the <b>if-feature</b> statement is only included if the specified feature is supported by the server. Otherwise, the object does not exist on the server.
<b>import</b> (by revision)	The import statement allows definitions from other modules to be used. A specific revision date can be used within the entire module. However, it is possible that different versions of imported typedefs and groupings can be used, if one imported module also imports some modules.
<b>include</b> (by revision)	The <b>include</b> statement provides the exact same features as the <b>import</b> statement, except it applied to sub-modules included within a module (or other sub-modules), instead of other modules. It allows multiple sub-modules to be combined to create one conceptual YANG module.
<b>key</b> <i>key-leaf-list</i> ;	The <b>key</b> statement indicates a set of one or more top-level leafs within the list that are used to name a specific instance of the particular list object. All protocol operations, such as <edit-config>, can be fully automated, based on the information in this statement.
<b>length</b> <i>length-spec-string</i> ;	The <b>length</b> statement is exactly like the <b>range</b> statement, except it limits the length of string <b>leaf</b> and <b>leaf-list</b> objects.
<b>mandatory</b> <i>boolean</i> ;	The <b>mandatory</b> statement indicates that the choice,

## Yuma Tools User Manual

	list or leaf must be provided by the client. It will not be created by the server. Most parameters are not mandatory however, so the default is 'false' if this statement is missing.
<b>max-elements</b> <i>number   'unbounded' ;</i>	Specifies the maximum number of instances that a <b>list</b> or <b>leaf-list</b> object can have in a valid database. The default is 'unbounded', if this statement is not present.
<b>min-elements</b> <i>number;</i>	Specifies the minimum number of instances that a <b>list</b> or <b>leaf-list</b> object must have in a valid database. The default is zero, if this statement is not present.
<b>must</b> <i>xpath-expr;</i>	If the object containing the <b>must</b> statement exists, then the XPath expression must evaluate to 'true' for the database to be valid. This provides referential integrity checks among related parameters.
<b>pattern</b> <i>pattern-string ;</i>	The <b>pattern</b> statement specifies a regular expression that must evaluate to 'true' in order for the corresponding string <b>leaf</b> or <b>leaf-list</b> object to be valid. Multiple patterns encountered in a nested typedef chain must all evaluate to 'true' for the object to be valid.
<b>range</b> <i>range-spec-string ;</i>	The <b>type</b> statement can specify the range of a numeric type. Since typedefs can be nested in YANG, the range statements are nested also, and constitute an AND expression (i.e., all the range tests must pass in the chain of type definitions.) THe keywords 'min' and 'max' indicate the minimum and maximum values from the parent typedef (if any), not the built-in type.
<b>refine</b> <i>refine-target-path { ... }</i>	The <b>refine</b> statement is defined within a <b>uses</b> statement, and allows the specific grouping to be customized for each individual copy of the grouping contents. The tools can automatically support the refined objects, based on the sub-statements within the <b>refine</b> statement.
<b>revision</b> <i>revision-date { ... }</i>	The <b>revision</b> statement identifies the most current version of a YANG module or sub-module. Multiple versions at once are supported in YANG.
<b>unique</b> <i>unique-node-list;</i>	The unique statement indicates an arbitrary tuple of descendant nodes within a list, which have to be unique within the list. These nodes are not keys, and can be nested anywhere within a single list entry.
<b>uses</b> <i>grouping-name;</i>	The <b>uses</b> statement inserts an instance of a reusable <b>grouping</b> , replacing the uses node within the conceptual data tree.
<b>when</b> <i>xpath-expr;</i>	The object containing the <b>when</b> statement is only allowed to exist if the XPath expression evaluates to 'true'. This provides a SPARSE AUGMENTS capability when combined with the augment

	statement.
--	------------

### 3.1.4 YANG LANGUAGE EXTENSIONS

There are several YANG extensions that are supported by Yuma. They are all defined in the YANG file named **yuma-ncx.yang**. They are used to 'tag' YANG definitions for some sort of automatic processing by Yuma programs. Extensions are position-sensitive, and if not used in the proper context, they will be ignored. A YANG extension statement must be defined (somewhere) for every extension used in a YANG file, or an error will occur.

Most of these extensions apply to **netconfd** server behavior, but not all of them. For example, the **ncx:hidden** extension will prevent **yangcli** from displaying help for an object containing this extension. Also, **yangdump** will skip this object in HTML output mode.

The following table describes the supported YANG language extensions. All other YANG extension statements will be ignored by Yuma, if encountered in a YANG file:

**YANG Language Extensions**

extension	description
<b>ncx:hidden;</b>	Declares that the object definition should be hidden from all automatic documentation generation. Help will not be available for the object in <b>yangcli</b> .
<b>ncx:metadata</b> “attr-type attr-name”;	Defines a qualified XML attribute in the module namespace. Allowed within an RPC input parameter. <b>attr-type</b> is a valid type name with optional YANG prefix. <b>attr-name</b> is the name of the XML attribute.
<b>ncx:no-duplicates;</b>	Declares that the <b>ncx:xsdlist</b> data type is not allowed to contain duplicate values. The default is to allow duplicate token strings within an <b>ncx:xsdlist</b> value.
<b>ncx:password;</b>	Declares that a string data type is really a password, and will not be displayed or matched by any filter.
<b>ncx:qname;</b>	Declares that a string data type is really an XML qualified name. XML prefixes will be properly generated by <b>yangcli</b> and <b>netconfd</b> .
<b>ncx:root;</b>	Declares that the container parameter is really a NETCONF database root, like <config> in the <edit-config> operations. The child nodes of this container are not specified in the YANG file. Instead, they are allowed to contain any top-level object from any YANG file supported by the server.
<b>ncx:schema-instance;</b>	Declares that a string data type is really an special schema instance identifier string. It is the same as an instance-identifier built-in type

	except the key leaf predicates are optional. For example, missing key values indicate wild cards that will match all values in <b>nacm</b> <dataRule> expressions.
<b>nacm:secure;</b>	Declares that the database object is a secure object. If the object is an <b>rpc</b> statement, then only the <b>netconfd</b> 'superuser' will be allowed to invoke this operation by default. Otherwise, only read access will be allowed to this object by default. Write access will only be allowed by the 'superuser', by default.
<b>nacm:very-secure;</b>	Declares that the database object is a very secure object. Only the 'superuser' will be allowed to access the object, by default.
<b>ncx:xslist</b> " <i>list-type</i> ";	Declares that a string data type is really an XSD style list. <b>list-type</b> is a valid type name with optional YANG prefix. List processing within <edit-config> will be automatically handled by <b>netconfd</b> .
<b>ncx&gt;xpath;</b>	Declares that a string data type is really an XPath expression. XML prefixes and all XPath processing will be done automatically by <b>yangcli</b> and <b>netconfd</b> .

### 3.1.5 YANG COMPILER

The Yuma programs all use the same centralized YANG language parser.

The complete YANG language is supported, as defined in the latest version (draft-ietf-netmod-yang-09.txt). The file naming conventions defined in this specification must be used, along with all the language definition rules.

Definitions can be contained in modules and/or sub-modules.

Any number of revisions of a module or submodule can be used concurrently. The **import-by-revision** and **include-by-revision** features of YANG are fully supported. Refer to the section 'Searching for Files' for more details.

All extension usage within YANG files is supported and saved. The application data is available to all Yuma programs, including netconfd server instrumentation. Refer to the 'YANG User Guide' for details on writing YANG files and using the extensions built into Yuma.

Note: The **smidump** is not part of Yuma, but it can be utilized to convert MIB modules written in SMIv2 into YANG modules, which can then be implemented in **netconfd**, and managed with **yangcli**. The freely available **libsmbi** library contains the **smidump** program.

### 3.1.6 YANG MODULE LIBRARY

The central system component is the set of YANG data model modules which define all available management information. This set of modules is expected to grow over time, and there is usually a high degree of reuse and inter-dependence between the modules.

## Yuma Tools User Manual

YANG modules can import other modules to reuse any definitions, and to augment objects in other modules. Each module represents one unique XML namespace used within the NETCONF protocol. A module can be partitioned into any number of submodules, each in a separate YANG file. The submodules are conceptually combined, and only the entire module is accessible to other modules.

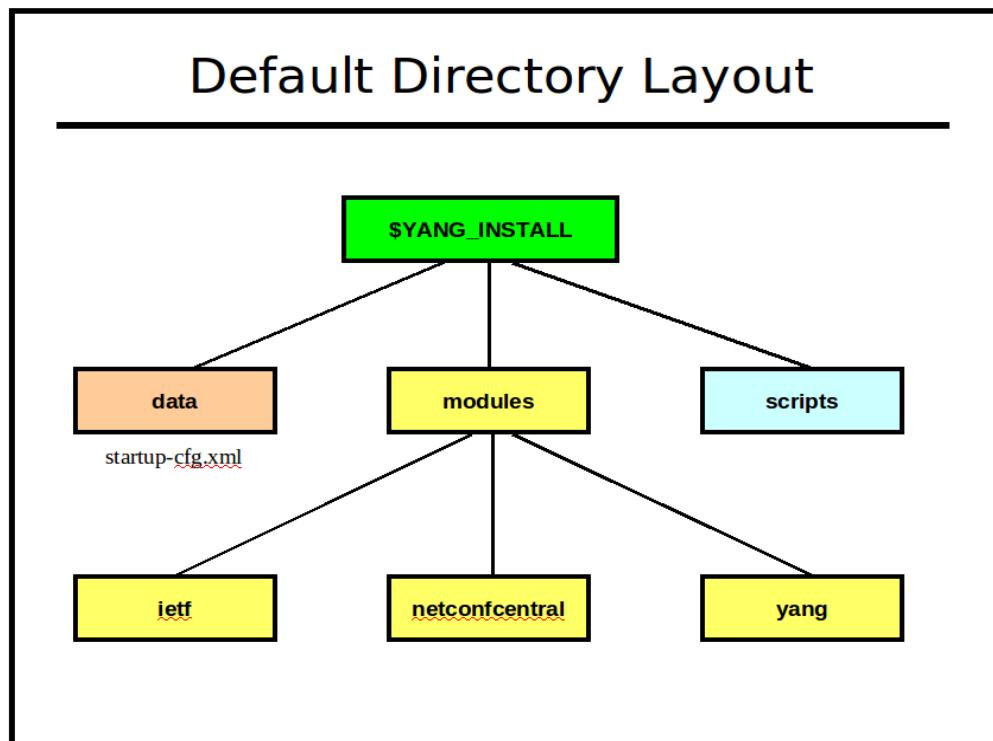
### **Directory Layout**

Yuma can utilize several directories to store files used during operation. By default, a 'root' directory and all of its sub-directories are searched for these files. Several different roots can be searched. Generally, there is one centralized root (`YUMA_INSTALL`) shared by all users, and one or more 'project' roots (`YUMA_HOME`), which can be shared but may belong to a single user.

The Yuma programs need to find and store the following types of files during operations:

- YANG modules and submodules (\*.yang):
- XML and text data files (usually \*.txt or \*.xml)
- command scripts for **yangcli**
- command-line-history file for **yangcli**

The search paths used to find these files are discussed in detail in the System Configuration section.



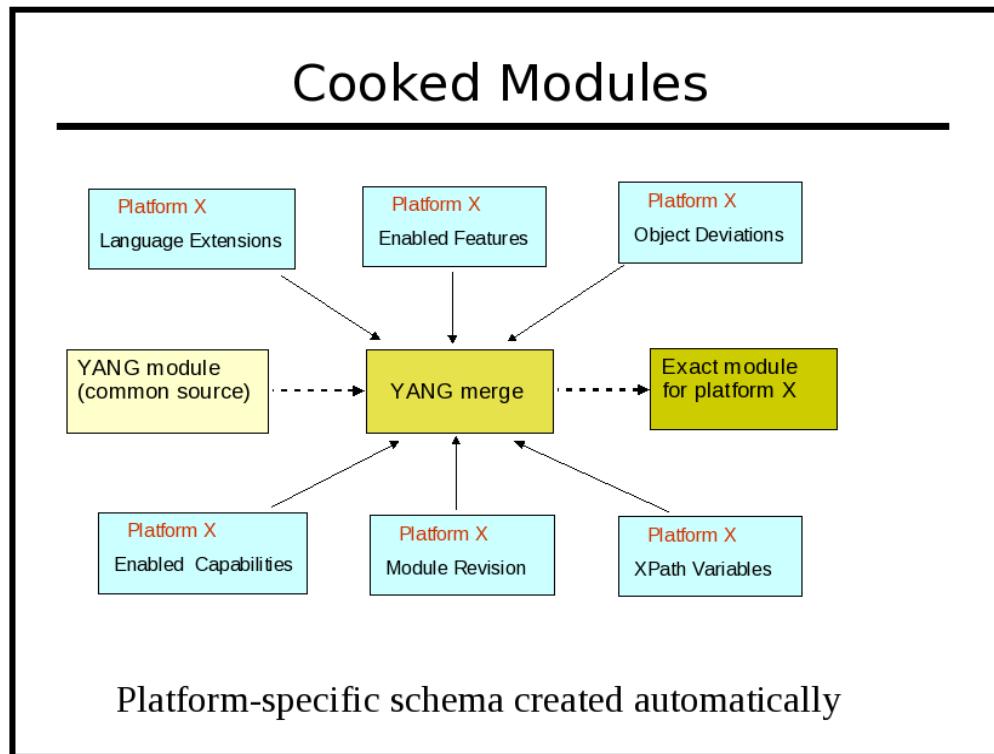
### **Module Revisions**

YANG has extensive module lifecycle support. Each module or submodule has a revision date, and multiple revisions of the same module or submodule may be used at once within the same server.

## Yuma Tools User Manual

The YANG module repository is the authoritative source of common management information for the **netconfd** server. However, different platform implementations of the same data model need to be 'adjusted' slightly to reflect differences in the feature support available on each platform.

Yuma has an extensive set of mechanisms to automate the maintenance of these platform-specific 'special requirements'. A single YANG module (plus 'patches' and deviations as needed for each platform) can be published, instead of a separate version of the YANG module for each platform.



## **Module Naming Conventions**

YANG module names are usually lower-case. Hyphen (-), underscore (\_) and period (.) characters are allowed, after the first character, which must be a letter. It is suggested that only the hyphen (-) character be used as a separator between module name string components. YANG files must use the suffix '.yang'.

There are two forms of YANG file names: with and without a revision date.

### **module.yang**

ietf-netconf-state.yang (no revision or unspecified revision)

### **module.revision-date.yang**

ietf-netconf-state.2009-04-17.yang (must be the 2009-04-17 version)

## Yuma Tools User Manual

These naming conventions are important when Yuma needs to resolve an 'import' or 'include' statement in a YANG file. Refer to section X.X for more details on YANG module search paths and the 'import-by-revision' feature of YANG.

### 3.1.7 YANG FILES

YANG modules and submodules are text files encoded in UTF-8. . There is also an alternate XML encoding called YIN. Sometimes the term YANG module is used to refer to the conceptual module, whether it is encoded in YANG format or YIN format.

All Yuma Tools programs will accept either encoding format, however line and column numbers are not correct in log messages for YIN encoded modules. Instead, each XML node is given a monotonically increasing value, and the XML document order is used instead of line numbers in error/warning messages for YIN files. The column number is always '1' for YIN files.

A module can be validated and checked for possible programming mistakes, by using the **yangdump** program. Many 'reports' can also be generated:

- exported symbols (--exports)
- imported modules (--dependencies)
- object identifiers (--identifiers)

The **yangdump** program is also used to generate other files, derived from the YANG content:

- **XML Schema Document (XSD)**: extends the NETCONF XSD with the YANG content layer definitions (--format=xsd)
- **HTML <div>** or full file output: hyper-linked, color-coded formatting of YANG modules to support netconf-central or other WEB-based documentation system. There are several options for configuring the output, and all formatting is done with Cascading style-sheets (CSS) (--format=html)
- **netconf-central** documentation SQL database input file: supports the automated online documentation of YANG content (--format=sqldb). Refer to the netconfcentral.sql file for details on this output, in the Developer Manual. This feature is only available in the SDK version of **yangdump**.
- **server instrumentation codestubs**: the instrumentation callback functions, used in **netconfd** for activating specific YANG content, can be generated. This procedure is described in more detail in the Developer Manual. This feature is only available in the SDK version of **yangdump**.
- **canonical YANG**: a YANG file can be reformatted so all statements are indented uniformly, and always appear in the same order. Objects marked as hidden (see the 'hidden' extension in yuma-ncx.yang) will not be generated. (--format=yang)
- **copy-YANG-and-set-name**: A YANG module can be validated and then copied (if no errors) to another location, adding the revision-date to the file name. (--format=copy)

### 3.1.8 NETCONF MANAGERS

The NETCONF client is an application that initiates and utilizes NETCONF sessions to control and monitor a NETCONF server.

Yuma includes the **yangcli** application for this purpose. It can be used as a stand-alone tool with any NETCONF server.

### 3.1.9 NETCONF AGENTS

## Yuma Tools User Manual

The NETCONF server is a server application that is always running on the managed device. It listens for NETCONF session requests from a NETCONF client, and allows specific users to access specific subsets of the available content (operations, database access, and notifications). It processes all incoming protocol operation requests from the client, and insulates all the instrumentation code from these protocol operations.

Yuma includes the **netconfd** application for this purpose. It can be run on several different platforms, or easily adapted to embedded platforms.

## 4 System Configuration

The Yuma programs use YANG to define its configuration parameters.

The 'ncx:cli' extension is used within a container with the same name as the program to define all CLI parameters. Some parameters are shared (see `yuma-app-common.yang`), so they are not located directly in the container.

```
container yangcli {
    ncx:cli;
    // yangcli CLI parameters defined as choices and leafs here
}
```

The following YANG modules are provided, which contain all the configuration parameters for Yuma:

- **yuma-types.yang**: contains common data types used in the Yuma applications
- **yuma-app-common.yang**: contains common CLI parameters used in all Yuma applications
- **yuma-ncx.yang**: contains YANG extensions used in any YANG module, including Yuma application modules
- **yangdump.yang**: configuration parameters for the **yangdump** application
- **yangdiff.yang**: configuration parameters for the **yangdiff** application
- **yangcli.yang**: configuration parameters and local commands for the **yangcli** application
- **netconfd.yang**: configuration parameters for the **netconfd** server

Note:

- The **netconf-subsystem** program does not have any configuration parameters at this time, so there is no YANG file defined for it.
- The **openssh** SSH server is configured separately, using the **sshd\_config** file.
- The **libtecla** library, used by the `yangcli` program for command line editing support, has its own configuration file `~/.tecla`, to override the default (emacs) editing key assignments.

Yuma applications can accept configuration parameters from 3 sources, checked in the following order:

1. environment variables
2. command line parameters
3. configuration file

### 4.1 Environment Variables

The Yuma programs utilize system environment variables to customize and simplify configuration and operation of the programs.

These environment variables typically specify file search paths or default directory locations.

## Yuma Tools User Manual

The following environment variables are used within Yuma:

- HOME
- YUMA\_HOME
- YUMA\_INSTALL
- YUMA\_MODPATH
- YUMA\_DATAPATH
- YUMA\_RUNPATH

### 4.1.1 \$HOME

The **\$HOME** environment variable contains the directory specification of the user's home directory, and is expected to be set by the system shell before use. The Yuma programs expect (by default) that sub-directories and files contained in this directory will be readable and writable.

Default value: none

CLI override: none

C shell example:

```
setenv $HOME /home/andy
```

Bash shell example:

```
set $HOME=/home/andy
export HOME
```

### 4.1.2 \$YUMA\_HOME

The **\$YUMA\_HOME** environment variable contains the directory specification of the current Yuma project root directory. This is the path to the 'netconf' directory, within a Yuma source tree.

Default value: none

CLI override: --yuma-home

CLI example:

```
--yuma-home=/home/andy/swdev/yuma/trunk/netconf
```

C shell example:

```
setenv $YUMA_HOME /home/andy/swdev/yuma/trunk/netconf
```

Bash shell example:

```
set $YUMA_HOME=/home/andy/swdev/yuma/trunk/netconf
export YUMA_HOME
```

## 4.1.3 \$YUMA\_INSTALL

The **\$YUMA\_INSTALL** environment variable contains the directory specification of the Yuma installation root directory.

Default value: /usr/share/yuma

CLI override: none

C shell example:

```
setenv $YUMA_INSTALL /sw/yuma
```

Bash shell example:

```
set $YUMA_INSTALL=/sw/yuma
export YUMA_INSTALL
```

## 4.1.4 \$YUMA\_MODPATH

The **\$YUMA\_MODPATH** environment variable contains a list of directory specifications that should be searched (in order) to find YANG or YIN modules and submodules. It can be used to extend the search path beyond the default locations.

The syntax for this parameter is a string containing the desired directory paths, separated by colon (:) characters. If the trailing forward slash (/) character is missing, then it will be added when searching for files.

By default, each entire directory and all its sub-directory contents will be searched for the requested file. This can be overridden with the **--subdirs** parameter. Refer to the Command Line Parameter Reference for more details. If **--subdirs=false** is used, then only the specified directory will be searched instead.

Note: This parameter specifies the exact directory locations when searching for files. This is different than the **\$HOME**, **\$YUMA\_HOME**, and **\$YUMA\_INSTALL** environment variables, which specify a Yuma root directory.

Default value: none

CLI override: --modpath

CLI example:

```
--modpath="$HOME/modules2:/usr/local/modules"
```

C shell example:

```
setenv $YUMA_MODPATH "$HOME/modules2:/usr/local/modules"
```

Bash shell example:

```
set $YUMA_MODPATH="$HOME/modules2:/usr/local/modules"
export YUMA_MODPATH
```

## 4.1.5 \$YUMA\_DATAPATH

The **\$YUMA\_DATAPATH** environment variable contains a list of directory specifications that should be searched (in order) to find data files used by Yuma applications. It can be used to extend the search path beyond the default locations.

Data files used by the **yangcli** program are affected by this environment variable.

The location where the **netconfd** program keeps the file **startup-cfg.xml** is also affected by this environment variable. This file contains the contents of the non-volatile <startup> database, which is loaded into the <running> database when the server boots.

The syntax for this parameter is a string containing the desired directory paths, separated by colon (:) characters. If the trailing forward slash (/) character is missing, then it will be added when searching for files.

By default, each entire directory and all its sub-directory contents will be searched for the requested file. This can be overridden with the **--subdirs** parameter. Refer to the Command Line Parameter Reference for more details. If **--subdirs=false** is used, then only the specified directory will be searched instead.

Note: This parameter specifies the exact directory locations when searching for files. This is different than the **\$HOME**, **\$YUMA\_HOME**, and **\$YUMA\_INSTALL** environment variables, which specify a Yuma root directory.

Default value: none

CLI override: --datapath

CLI example:

```
--datapath="$HOME/modules2:/usr/local/modules"
```

C shell example:

```
setenv $YUMA_MODPATH "$HOME/modules2:/usr/local/modules"
```

Bash shell example:

```
set $YUMA_MODPATH="$HOME/modules2:/usr/local/modules"
export YUMA_MODPATH
```

## 4.1.6 \$YUMA\_RUNPATH

The **\$YUMA\_RUNPATH** environment variable contains a list of directory specifications that should be searched (in order) to find script files used by Yuma applications. It can be used to extend the search path beyond the default locations.

Script files used by the **yangcli** program are affected by this environment variable.

The syntax for this parameter is a string containing the desired directory paths, separated by colon (:) characters. If the trailing forward slash (/) character is missing, then it will be added when searching for files.

By default, each entire directory and all its sub-directory contents will be searched for the requested file. This can be overridden with the **--subdirs** parameter. Refer to the Command Line Parameter

## Yuma Tools User Manual

Reference for more details. If **--subdirs=false** is used, then only the specified directory will be searched instead.

Note: This parameter specifies the exact directory locations when searching for files. This is different than the **\$HOME**, **\$YUMA\_HOME**, and **\$YUMA\_INSTALL** environment variables, which specify a Yuma root directory.

Default value: none

CLI override: --runpath

CLI example:

```
--runpath="$HOME/scripts:/usr/local/scripts"
```

C shell example:

```
setenv $YUMA_RUNPATH "$HOME/scripts:/usr/local/scripts"
```

Bash shell example:

```
set $YUMA_RUNPATH="$HOME/scripts:/usr/local/scripts"  
export YUMA_RUNPATH
```

## 4.2 Searching for YANG Files

All Yuma programs search for YANG files in the same manner, using the same configuration parameters. The current working directory is included in this search path, so it is important to consider the directory in which a Yuma program is invoked. The search ends as soon as a suitable matching file is found.

There are two types of module searches:

1. searches on behalf of configuration parameters
2. searches on behalf of YANG import or include statements

The first term in a path specification may contain special character sequences:

- If the first character is the forward slash ('/'), then the entire path specification is used as an absolute path specification.  
`/usr/share/yang/modules`
- If the first character is not the forward slash ('/'), and no special characters are found instead, then the entire path specification is used as an relative path specification, starting from the current working directory.  
`./more-modules/test7.yang  
./this-dir/my-module.yang  
testmodule.yang  
old-modules/version7/`
- If the first character is the tilde ('~') character, followed by the forward slash ('/') character, then the file search will start in the current user's \$HOME directory .  
`~/modules/test/test.yang`

## Yuma Tools User Manual

- If the first character is the tilde ('~') character, followed by a user name, and then the forward slash ('/') character, then the file search will start in the specified user's \$HOME directory . If the user is unknown, then the path specification is invalid.

```
~andy/modules/test/test.yang  
~fred/scripts
```

- If the first character is the dollar sign ('\$') character, followed by an environment variable name, and then the forward slash ('/') character, then the file search will start in the directory indicated by the contents of the environment variable. If the variable is unknown, or its contents do not represent a valid directory location, then the path specification is invalid.

```
$WORKDIR/tests/test-all-interfaces  
$YUMA_HOME/data/startup-cfg.xml
```

Note: Whenever Yuma searches a directory, it checks for the expected file type, but ignores the following:

- all files and sub-directories that begin with the period (.) character
- any directory named 'CVS'
- symbolic links for regular files

The following environment variables affect file searches:

- \$HOME
- \$YUMA\_HOME
- \$YUMA\_MODPATH
- \$YUMA\_DATAPATH
- \$YUMA\_RUNPATH

The following configuration parameters affect file searches:

- --yuma-home
- --modpath
- --datapath
- --runpath
- --subdirs

### 4.2.1 PARAMETER SEARCHES

A parameter search is started on behalf of a CLI parameter, such as the **--module** parameter, used by the **yangdump** program. A search of this type can include directory path and file extension in the search parameter. If a filename with a file extension (must be '.yang') is given, then only that exact file will be checked. The current working directory will be used in this case, if no directory path (or a relative directory path) is provided.

```
--module=test.yang  
--module=../more-modules/test3.2009-04-01.yang
```

If the exact filename is not found, then the search failed.

If a parameter based search does not have any directory path or file extension fields present, then a parameter search is the same as an import/include search.

## 4.2.2 IMPORT/INCLUDE SEARCHES

An import or include search is started on behalf of a YANG 'import' or 'include' statement. A search of this type includes only the module or submodule name, with no directory or file extension present. An optional 'revision-date' statement can be used in YANG, which means only a version of the YANG file with that exact current revision date will be used.

There are separate search algorithms, depending on whether the revision-date is used in the YANG import or include statement, and whether the imported or included module has a current revision statement.

### **Mode 1: import-by-revision**

In this example, an import statement is causing a search for a module named 'foo' with a revision date of '2009-01-15'.

If a revision-date is used in the import or include statement, then the module search path will be checked as follows:

First, find a file with the same revision-date in the file name:

```
import foo {
    revision-date "2009-01-15";
    prefix foo;
}
```

If the file 'foo.2009-01-15.yang' is found, and the current revision statement in the module is equal to '2009-01-15', then the search is successfully terminated.

```
// file foo.2009-01-15.yang
module foo {

    namespace "http://example.com/ns/foo";
    prefix foo;

    // rest of header follows

    revision 2009-01-15 {
        description "Initial version.";
    }

    // rest of module follows
}
```

If the file is not found, or the most current revision date is not correct, then the module search is repeated for 'foo.yang'. If the file 'foo.yang' is found, and the current revision statement in the module is equal to '2009-01-15', then the search is successfully terminated.

```
// file foo.yang
```

## Yuma Tools User Manual

```
module foo {  
  
    namespace "http://example.com/ns/foo";  
    prefix foo;  
  
    // rest of header follows  
  
    revision 2009-01-15 {  
        description "Initial version.";  
    }  
  
    // rest of module follows  
}
```

If the file is not found, or the most current revision date is not correct, then the module search failed.

### **Mode 2: import any revision**

If no file name with the specified revision-date value is found, then the module search path is checked for a file with no revision-date in the file name:

```
import foo {  
    prefix foo;  
}
```

If the file 'foo.yang' is found, then it is used, regardless of the most current revision date (if any) found in the module. If it is not found then the module search failed.

Note: The first instance of 'foo.yang' in the module search path will be used, even if a more current version is available, later in the search path.

### **4.2.3 FILE SEARCH PATHS**

Yuma uses configurable search paths to find the various files that are needed during operation.

#### **Module Search Path**

- If the module parameter is specified with a path or file suffix, the that filespec is tried, relative to the current working directory. If it is not found, or not the correct revision date, then the search terminates in failure.

```
--module=../test.yang
```

## Yuma Tools User Manual

- If the module is specified without any path or file extension fields, then the module search path is checked, in order. The first step which produces a match terminates the search successfully. If all steps are exhausted and no match is found then the search terminates in failure.

```
--module=foo
```

1. The current working directory is checked. No sub-directories are checked, if any are present.
2. Each directory specified in the **\$YUMA\_MODPATH** environment variable, or set with the **-modpath** configuration parameter, is checked.
  - If the **--subdirs=false** parameter is set, then only each top-level directory will be checked. If it is not set, then sub-directories will be searched.
3. The **\$HOME/modules** directory is checked.
  - If the **--subdirs=false** parameter is set, then only each top-level directory will be checked. If it is not set, then sub-directories will be searched.
4. The **\$YUMA\_HOME/modules** directory is checked.
  - If the **--subdirs=false** parameter is set, then only each top-level directory will be checked. If it is not set, then sub-directories will be searched.
5. The **\$YUMA\_INSTALL/modules** directory is checked.
  - If the **--subdirs=false** parameter is set, then only each top-level directory will be checked. If it is not set, then sub-directories will be searched.

### **Data Search Path**

Yuma programs can store data used during operation.

An example of a data file is the startup configuration file used by **netconfd**, usually called **startup-cfg.xml**.

1. If the file name has an absolute path specification, then that exact file location is tried. If no match is found, then the search will terminate in failure.
2. The current working directory is checked. No sub-directories are checked, if any are present.
3. Each directory specified in the **\$YUMA\_DATAPATH** environment variable, or set with the **-datapath** configuration parameter, is checked.
  - If the **--subdirs=false** parameter is set, then only each top-level directory will be checked. If it is not set, then sub-directories will be searched.
2. The **\$HOME/data** directory is checked.
  - If the **--subdirs=false** parameter is set, then only each top-level directory will be checked. If it is not set, then sub-directories will be searched.
3. The **\$YUMA\_HOME/data** directory is checked.
  - If the **--subdirs=false** parameter is set, then only each top-level directory will be checked. If it is not set, then sub-directories will be searched.

## Script Search Path

The **yangcli** program can store script files used during operation.

1. If the file name has an absolute path specification, then that exact file location is tried. If no match is found, then the search will terminate in failure.
2. The current working directory is checked. No sub-directories are checked, if any are present.
3. Each directory specified in the **\$YUMA\_RUNPATH** environment variable, or set with the **-runpath** configuration parameter, is checked.
  - If the **--subdirs=false** parameter is set, then only each top-level directory will be checked. If it is not set, then sub-directories will be searched.
4. The **\$HOME/scripts** directory is checked.
  - If the **--subdirs=false** parameter is set, then only each top-level directory will be checked. If it is not set, then sub-directories will be searched.
5. The **\$YUMA\_HOME/scripts** directory is checked.
  - If the **--subdirs=false** parameter is set, then only each top-level directory will be checked. If it is not set, then sub-directories will be searched.
6. The **\$YUMA\_INSTALL/scripts** directory is checked.
  - If the **--subdirs=false** parameter is set, then only each top-level directory will be checked. If it is not set, then sub-directories will be searched.

## 4.3 Configuration Files

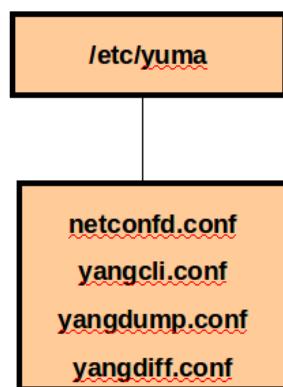
The Yuma program configuration parameters can be stored in text or XML files.

The **--config** parameter is used to specify that configuration parameters should be retrieved from a file instead of the command line.

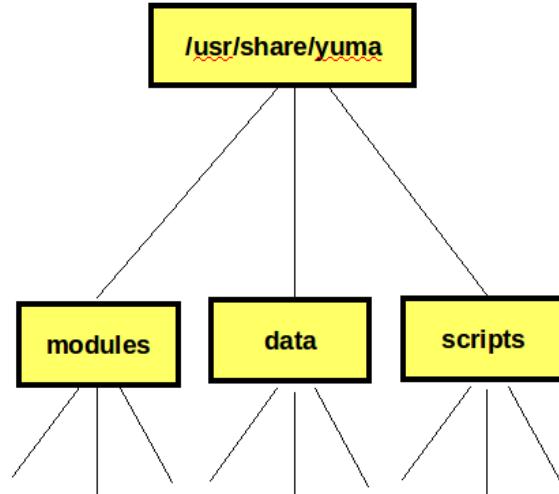
Any other configuration parameter (except **--config**) can be stored in a configuration file used for program input.

# Default Linux Data Directories

CLI config files



YANG, XML, other files



## 4.3.1 XML CONFIGURATION FILES

The XML format for these files follows the structure of the NETCONF <config> element. Each parameter is stored within a container identifying the application which it is being configured. The **netconfd** stores its non-volatile <startup> database in this format. XML configuration file contents can appear in any order.

The following configuration parameters affect the generation and display of XML configuration files by **netconfd**:

- --indent
- --with-defaults

The following configuration parameter affects the location of XML configuration files by **netconfd**:

- --datopath
- \$YUMA\_DATAPATH environment variable

## Yuma Tools User Manual

Note : The IETF may standardize this container format soon. Do not rely on the top-level namespace URI. Any top-level element name <config>, in any namespace (even none), should be expected to contain a complete NETCONF database, or a subset of a NETCONF database.

The following example show some database objects from the NETCONF Access Control Model (yuma-nacm.yang), in XML configuration file format.

```
// file startup-cfg.xml
<?xml version="1.0" encoding="UTF-8"?>
<nd:config xmlns:nd="http://netconfcentral.com/ns/netconfd">
  <nacm:nacm xmlns:nacm="http://netconfcentral.com/ns/nacm">
    <nacm:groups>
      <nacm:group>
        <nacm:groupIdentity>nacm:admin</nacm:groupIdentity>
        <nacm:userName>andy</nacm:userName>
        <nacm:userName>fred</nacm:userName>
        <nacm:userName>barney</nacm:userName>
      </nacm:group>
    </nacm:groups>
    <nacm:rules>
      <nacm:moduleRule>
        <nacm:moduleName>netconf</nacm:moduleName>
        <nacm:allowedRights>read write exec</nacm:allowedRights>
        <nacm:allowedGroup>nacm:admin</nacm:allowedGroup>
      </nacm:moduleRule>
    </nacm:rules>
  </nacm:nacm>
</nd:config>
```

### 4.3.2 TEXT CONFIGURATION FILES

The Yuma text configuration file format is based on some common Unix .conf file formats:

- A hash mark until EOLN is treated as a comment

```
# this is a comment
log-level info      # this is also a comment
```

- All text is case-sensitive
- Whitespace before or within a line is not significant
- The 'end a line' (EOLN) character ('\n') is used to end a command, so whitespace at the end of a line is significant.

## Yuma Tools User Manual

- To enter a command on multiple lines, use an escaped EOLN (backslash-EOLN) for all but the last line

```
this is a command line
this is the start \
of a long \
three line command
this is a new command
```

- A YANG container parameter is represented by the container name, followed by a left curly brace ('{'), zero or more child nodes, and then a right curly brace ('}').

```
yangdump {
    # set some display control parameters
    log-level debug2
    warn-linelen 72
    indent 4
}
```

- A YANG list parameter is represented by the list name, followed by a whitespace separated sequence of key leaf values, followed by a left curly brace ('{'), zero or more child nodes, and then a right curly brace ('}').

```
ifStackEntry 11 42 {
    # the key leafs will also printed here
    ifStackHigherLayer 11
    ifStackLowerLayer 42
    ifStackStatus active
}
```

- Configuration files which are used with command line parameters may include program parameters for multiple applications.
  - Only the top-level container that matches the name of the program will be used.
  - Any other top-level containers will be ignored
  - Only the first instance of the desired program container will be used. Any additional containers will be ignored.

```
// test.conf
yangdump {
    # common yangdump parameters here
}

yangdiff {
```

```
# common yangdiff parameters here
}
```

- Configuration file parameters can appear in any order. Only list index strings need to appear in their defined order.
- The following configuration parameters affect generation and display of text configuration files
  - --indent
  - --with-defaults
  - --display-mode

## 4.4 Bootstrap CLI

Since Yuma programs use YANG to define CLI parameters, there needs to be an initial bootstrap CLI phase, in order to process parameters which affect the way YANG files are processed.

The bootstrap CLI is not as flexible as the main CLI processor, and the syntax is more strict. Parameters must be entered in either of the following forms:

- --name
- --name=value

If parameters are not entered in this format, then they will be skipped until the main CLI parameter processing is done. This may cause undesirable changes in key parameters, such as the module search path.

The following configuration parameters are also bootstrap parameters, and will take effect immediately, if entered from the command line:

- **--log**: log messages to the specified file instead of STDOUT
- **--log-level**: set the logging verbosity level
- **--log-append**: use the existing log file (if any) instead of overwriting it
- **--modpath**: use the specified module search path. This will override the **\$YUMA\_MODPATH** environment variable, if it is set
- **--yuma-home**: use the specified project root. This will override the **\$YUMA\_HOME** environment variable, if it is set

Refer to the Yuma CLI Reference for more details. on these configuration parameters.

## 4.5 Configuration Parameters

Command line parameters are used to provide input to Yuma programs when they are invoked. They are also used extensively by the **yangcli** program, to represent RPC method input parameters and database nodes which are part of NETCONF operation content, such as the **<config>** parameter within the **<edit-config>** operation.

### 4.5.1 PARAMETER SYNTAX

A CLI parameter has 2 forms:

- Parameter contains a YANG type of 'empty' or a zero-length 'string':

- <prefix><parameter-name>
- Everything else:  
<prefix><parameter-name><separator><value>

There are up to 4 components in a CLI parameter:

1. **prefix**: consists of 0, 1, or 2 consecutive dash characters.
2. **parameter name**: name of the parameter. A partial name may be used if it is unique.
3. **separator**: either consists of the 'equals sign' character ('='), which may be preceded or followed by whitespace, or just whitespace with no equals sign character.
4. **value**: a quoted or unquoted string, an empty string is only allowed if quotes are entered.

The following example shows some ways the leaf 'foo' could be entered as a CLI parameter:

```
leaf foo {  
    type uint32;  
}  
  
foo=7  
-foo=7  
--foo=7  
--foo =7  
foo 7  
-foo 7  
-foo = 7  
--foo 7  
--foo "7"  
foo 7
```

### 4.5.2 ncx:cli EXTENSION

The **ncx:cli** extension is used in YANG container definitions, which represent the program CLI parameters, not NETCONF database parameters. It does not take any parameters, and is defined in **yuma-ncx.yang**.

```
container yangcli {  
    ncx:cli;  
  
    // all the yangcli CLI parameters  
}
```

## Yuma Tools User Manual

If this extension is present, then **netconfd** will ignore the container when loading the database object definitions. Only the program with the same name as the container will use the CLI parameter definition.

### 4.5.3 NCX:DEFAULT-PARM EXTENSION

The **ncx:default-parm** extension is used within a container with an **ncx:cli** extension, or within an 'input' section of an RPC operation definition. It is defined in **yuma-ncx.yang**.

If no parameter name is found when processing CLI parameter input, and the **ncx:default-parm** extension is present in the container or RPC input being processed, then the specified parameter name will be used instead of generating an error. The value must be valid for the parameter syntax, according to its YANG definition. This means that for the default parameter, only the <value> component of the complete parameter syntax may be used, as well as the normal forms.

```
container yangdump {
    ncx:cli;
    ncx:default-parm module;

    // all the yangdump CLI parameters
}
```

When invoking the **yangdump** program, the default CLI parameter is **--module**. These two command lines are equivalent:

```
yangdump --module=test1 --module=test2
yangdump test1 test2
```

A string that does not start with any dashes will still be tried as a parameter name, before trying the default parameter. If the value used for a default parameter conflicts with another parameter name, then the normal form must be used, instead of this form.

```
yangdump log-app test1
```

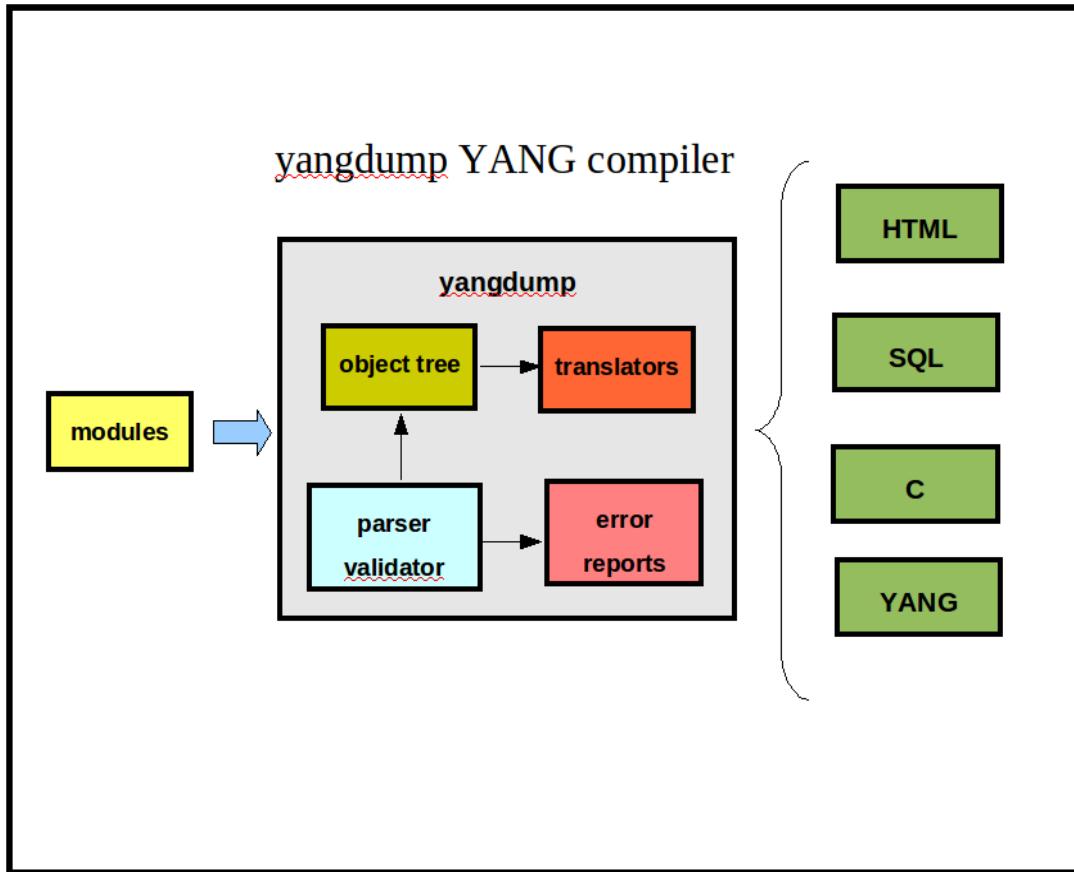
Even if there was a module named 'log-app', it would not be tried as a **--module** parameter, since it also matches the **--log-append** parameter.

Note: the default parameter form can be used in conjunction with shell wildcard characters, depending on the shell.

```
yangdump *.yang
yangdump --subtree=.
```

These commands are equivalent in the **yangdump** program.

## 5 yangdump User Guide



### 5.1 Introduction

The **yangdump** program is used to validate YANG files, and convert them to other formats. It is normally used by authors while developing new YANG data models. The compiler and validation engine portion of **yangdump** (libncx.so) is used in all **Yuma** programs.

#### 5.1.1 FEATURES

The **yangdump** compiler has the following features:

- full support for all YANG language constructs.
- full support for YANG submodules.
- full support for YANG revisions and deviations across an arbitrary set of YANG files.
- validates the entire YANG source file, even statements within unused objects.

## Yuma Tools User Manual

- builds a complete internal representation of the 'cooked' object tree, including deviations, to validate or output the exact data structures that result from a set of YANG files.
- Over 100 separate validation checks, errors, and warnings, covering every type of programming mistake that can be made in a YANG file.
- Some warnings are fully configurable, and any warning can be suppressed.
- Can process any number specific modules and directory trees containing modules at once.
- Any number of modules containing deviations can be specified, and any target nodes in all modules affected will be properly patched.
- Ignores CVS and subversion directories, if encountered in directory searches.
- Attempts to find all errors and warnings possible in the file, instead of stopping on the first error.
- Verifies that all YANG statement syntax is correct, according to the specification.
- Verifies that all YANG typedef statements are correct:
  - The entire typedef chain is checked, until a built-in data type is reached.
  - Checks valid combined refinements to named types and built-in types
  - Check for valid range statements.
  - Check for valid pattern statements.
  - Verify the path statement is correct for leafref type statements.
  - Verify the default statement is valid.
- Compile time loop detection:
  - import statement loops.
  - include statement loops.
  - belongs-to statement references self.
  - typedef type statement references self.
  - derived type statement loops.
  - path statement loops for leafref objects.
  - base statement loops for identity statements.
  - feature name reference loops for if-feature statements.
  - uses statement loop in groupings.
- Compile-time duplicate detection:
  - import statements
  - include statements
  - must statements
  - if-feature statements
  - unique statements
  - local typedef of grouping name collision
  - verifies that the correct number of instances appear for every YANG sub-statement.
  - duplicate sibling node names due to uses or augment statement expansion
  - checks multiple refine statements within a uses statement for duplicate refinements to the same target node.

## Yuma Tools User Manual

- checks multiple deviate statements within a deviation statement for duplicate alterations to the same target node.
- checks deep within all choice statements to make sure that no accessible object names conflict with sibling nodes that will appear in the value nodes, within a NETCONF database.
- Verifies that all default statements are correct for the declared type.
  - Only the static properties of the leafref and instance-identifier built-in data types can be validated at compile time.
- Verifies that all XPath expressions in must and when statements, even those in groupings, contain only valid XPath and reference valid data nodes.
- Verifies all when statements for a given cooked object, even those inherited from uses and augment statements.
- Detects namespace statement conflicts across a set of modules.
- Detects prefix statement conflicts across a set of modules.
- Detects augment statement conflicts.
- Checks if any key leafs are more conditional than their parent node. This can occur if any 'when' or 'if-feature' statements apply to the key leaf, but not to the parent.
- Detects unused typedefs and groupings.
- Checks line length and identifier length.
- Checks revision statements present and in correct order.
- Detects any top-level mandatory configuration objects:
  - leaf with mandatory statement set to true.
  - choice with mandatory statement set to true.
  - non-presence container with any mandatory descendants.
- Checks all refine statements for proper usage, and checks that the resulting set of objects remains valid. Multiple refine statements for the same target will be combined and checked as if there was only one refine statement for each target node.
- Checks all deviation statements for proper usage, and checks that the resulting set of objects remains valid. Multiple deviation statements for the same target will be combined and checked as if there was only one deviation statement for each target node.

The **yangdump** translator has the following features:

- Full support for YANG string specification syntax:
  - All double quoted string formatting rules.
  - Preservation of single-quoted string content.
  - All forms of string concatenation.
  - Unquoted string without any whitespace.
- Generates YIN syntax from the YANG token sequence
  - YIN format is the standard XML version of a YANG module
- Generates hyper-linked HTML for a set of YANG modules.
  - Rich set of configuration parameters to control HTML generation.
  - Uses configurable and customizable Cascading Style Sheets (CSS).

## Yuma Tools User Manual

- Can generate full WEB page or single <div> for embedding in a WEB page.
- Can combine all sub-modules into a single conceptual module.
- Objects tagged as **ncx:hidden** will be ignored in the HTML output.
- Generates canonical YANG (all statements in the same order, etc.):
  - Combines refine and deviation statements for the same target.
  - Combines range statement components into canonical form.
  - Generates consistent (configurable) indentation for all statements.
  - Objects tagged as **ncx:hidden** will be ignored in the YANG output.
- Generates SQL statements for populating the netconf-central database with quick lookup information about a set of YANG modules. This feature is only available in the SDK version of **yangdump**.
- Generates C source code stubs for **netconfd** server instrumentation libraries, for dynamic loading of a YANG module. This feature is only available in the SDK version of **yangdump**.
- Generates informational reports on the contents of a YANG file:
  - Imported modules (dependencies).
  - Exported symbols (exports).
  - Object name tree (identifiers)
  - Module revision date (modversion)

The following features are not yet available, but planned for a future release:

- YANG to YIN translation
- YANG module metrics reports
- naming conflicts report
- preserve and translate YANG comments
- preserve or reformat YANG string concatenation

### 5.1.2 STARTING YANGDUMP

The current working directory in use when **yangdump** is invoked is important. It is most convenient to run **yangdump** from a work directory, rather than the installation directory or within the module library.

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

- To get the current version and exit:

```
yangdump --version
```

- To get program help and exit:

```
yangdump --help
yangdump --help --brief
yangdump --help --full
```

## Yuma Tools User Manual

The default parameter for the **yangdump** CLI is the --module parameter. If an unknown parameter is given and it could be a valid module parameter, then it will be treated as an instance of this parameter.

- To validate a single YANG module named 'foo', the following command lines are equivalent:

```
yangdump foo
```

```
yangdump --module=foo
```

- To validate the './test1' and './test2' directory subtrees as an entire set of modules, and save the output to 'mylogfile':

```
yangdump --subtree=test1 --subtree=test2 --logfile=mylogfile
```

- To get all the configuration parameters from a text file named '~/.yangdump-project1.conf':

```
yangdump --config=~/.yangdump-project1.conf
```

- To generate YANG HTML documentation files in the '~/work' directory (with the default naming scheme) for all the YANG modules in the './test1' directory subtree:

```
yangdump --subtree=test1 --output=~/work --format=html --defnames=true
```

### 5.1.3 STOPPING YANGDUMP

There is no interactive mode for **yangdump**, so there is no need for a command to exit the program.

The Control C character sequence can be used to cancel the **yangdump** processing in progress. However, this will leave any partially completed output files in place.

### 5.1.4 CONFIGURATION PARAMETER LIST

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

#### yangdump CLI Parameters

parameter	description
--config	Specifies the configuration file to use for parameters.
--datopath	Sets the data file search path.
--defnames	Specifies if the default naming scheme is used for any output files.
--dependencies	Generate the module dependencies report.

## Yuma Tools User Manual

--deviation	Specifies one or more YANG modules to load as deviations.
--exports	Generate the module exports report.
--format	Specifies the type of translation format that should be used for the output.
--help	Get context-sensitive help, then exit.
--help-mode	Adjust the help output (--brief, or --full).
--html-div	For HTML translation, controls whether to generate a <div> element instead of a complete HTML document.
--html-toc	For HTML translation, controls the table of contents that is generated.
--identifiers	Generate the module object identifiers report.
--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.
--modpath	Sets the module search path.
--module	Specifies one or more YANG modules to load upon startup.
--modversion	Generate the module version report.
--subdirs	Controls whether sub-directories will be searched during file searches.
--versionnames	Controls whether the revision date is used in YANG module file names.
--objview	Specifies whether raw or cooked objects will be generated during HTML and YANG translation.
--output	Specifies where output files should be generated.
--runpath	Sets the executable file search path.
--show-errors	Print all the error and warning messages, then exit.
--simurls	Controls how URL parameters are generated during HTML file output.
--subdirs	Controls whether sub-directories are searched for YANG files.
--subtree	Specifies one or more directory subtrees to search for YANG modules.
--unified	Controls whether to combine submodules into the main module in the translation output.
--urlstart	Specifies the start of URLs to use during HTML file generation.
--version	Prints the program version and then exit.
--warn-idlen	Controls how identifier lengths are checked.
--warn-linelen	Controls how line lengths are checked.

--warn-off	Suppresses the specified warning number.
--xsd-schemaloc	Generate schemaLocation attributes during XSD translation.
--yuma-home	Specifies the <b>\$YUMA_HOME</b> project root to use when searching for files.

## 5.2 Validating YANG Files

The **yangdump** program will always validate the modules specified in the configuration parameters, unless one of the quick exit modes is requested (--version or --help). If a valid **--format** parameter is present, then some sort of translation will also be attempted.

The modules or submodules to validate are specified with the **--module** and/or **--subtree** configuration parameters.

All sub-modules that are included, and all modules that are imported, are also validated. This is done in the order the import or include statements are encountered. Errors and warnings that occur in included submodules or imported modules may be repeated, if it is used more than once within the requested set of files to validate.

The **netconfd** server will refuse to run if any of its core YANG modules contain any errors, as reported by the **yangdump** parser.

The **--deviation** parameter is used to specify any YANG module files that contain YANG deviation statements, which may apply to any of the modules specified with the **--module** or **--subtree** parameters. Modules parsed as deviation files are not validated. Imported modules are not actually read, and therefore not validated either.

A module can appear in the deviation list and the module or subtree module list. The deviation statements will simply be processed separately from the other YANG statements found in the file.

If all the deviation statements for a module appear in the same module as the objects that are the target(s) of the deviations, then the **--deviation** parameter does not need to be used.

There is no way to specify the exact revision date associated with each file, at this time. The module search path needs to be configured such that the module search algorithms will find the appropriate versions.

This only applies if the revision-date statement is not present in the import statement, of course. Otherwise, only the specified revision-date will be used, or an error will be generated if the exact import file was not found.

### 5.2.1 YANGDUMP VALIDATION MESSAGES

A error is generated, and the error count for the module is incremented, if any violation of the YANG standard is detected.

A warning is generated, and the warning count for the module is incremented, if:

- any suspected misuse of the YANG standard is detected.
- any YANG statements which have no affect are detected.
- any duplicate statements which may conflict with previous statements are detected.
- any violation of a YANG usage guideline is detected.

## Yuma Tools User Manual

- any statements which may cause operational conflicts, due to duplicate values from another module is detected. The module prefix or an augment statement which adds a node with the same name are examples of this type of operational conflict.

An informational message is generated if any potentially interesting conditions or abnormality is detected.

A debugging message is generated if available, to track the internal behavior of the **yangdump** parser.

The default **--log-level** configuration parameter value is 'info'. It is strongly suggested that the log-level not be set to 'off' or 'error'. Instead, set the log level to at least 'warning', and use the **--warn-idlen**, **----warn-linelen**, and **--warn-off** configuration parameters to suppress or modify specific warning messages.

Generally, a context-specific error message is generated, followed by a structured error message.

A context-sensitive error message will begin "Error: ....".

A structured error message will begin with a line in the following format:

**<module>:<line-number>:<column-number>: error(<error-number>):<msg>**

where:

- module: module or submodule name

### 5.2.2 VALIDATION EXAMPLE

The following example shows the error messages that are generated for the 'testloops.yang' file, located in the modules/test sub-directory.

```
*** Generated by yangdump 0.9.7.452 at 2009-08-27T23:56:21Z

Error: import 'testloops' for current module
testloops.yang:6.5: error(327): import loop

Error: include 'testloops' for current submodule
testloops.yang:10.5: error(328): include loop

Error: typedef 'typeloop' cannot use type 'typeloop'
testloops.yang:35.8: error(325): definition loop detected

Error: named type loops with type 'typeloop2' on line 43
testloops.yang:47.8: error(325): definition loop detected

Error: uses of 'grouploop' is contained within that grouping, defined on line 50
testloops.yang:51.8: error(325): definition loop detected

Error: grouping 'grouploop2' on line 54 loops in uses, defined in module testloops, line 63
testloops.yang:54.5: error(325): definition loop detected

Error: leafref path in leaf LR1 loops with leaf LR3
```

```
testloops.yang:16.5: error(359): leafref path loop

Error: leafref path in leaf LR2 loops with leaf LR1
testloops.yang:22.5: error(359): leafref path loop

Error: leafref path in leaf LR3 loops with leaf LR2
testloops.yang:28.5: error(359): leafref path loop

*** /home/andy/modules/test/testloops.yang: 9 Errors, 0 Warnings
```

## 5.3 Translating YANG to Other Formats

The **yangdump** program can be used to translate YANG files to other formats, using the **--format** parameter. This section describes the available translation modes.

In all cases:

- the **--subtree** or **--module** parameter is required to specify the input files. Any number of these parameters can be entered, given in any order.
- the **--deviation** parameter will affect how objects are generated, if **--objview=cooked**.
- the **--output** parameter is usually specified to cause the files to be copied to a different directory. If this parameter is not present, then the current directory will be used for generation of output files. The default is either STDOUT, or to the current directory if **--defnames** is set to 'true'.
- the **--indent** parameter can be used to specify the number of spaces to indent each new nest level. The default value is 3 spaces.
- the **--defnames** parameter can be used to force the default naming scheme. For many translation modes, this parameter is the assumed default, and cannot be changed.
- the **--urlstart** parameter can be used to specify the string to start all URLs, if URLs are generated in the output.

### 5.3.1 YIN FORMAT

The standard YIN format can be generated with the **--format=yin** parameter value.

The YIN representation of a YANG module is an XML instance document consisting of a <module> or a <submodule> element.

The set of YANG prefixes used in the module will be used as the XML prefixes available in the document. These namespace attributes are added to the top-level element for the module prefix and any imported modules.

The following example shows the XML instance document that is generated for **--module=test4** and **--format=yin**:

```
<?xml version="1.0" encoding="UTF-8"?>
<module
  name="test4"
```

## Yuma Tools User Manual

```
xmlns="urn:ietf:params:xml:ns:yang:yin:1"
xmlns:t4="http://netconfcentral.com/ns/test4">
<namespace uri="http://netconfcentral.com/ns/test4" />
<prefix value="t4" />
<revision date="2009-09-09">
    <description>
        <text>
            Initial revision.
        </text>
    </description>
</revision>
<typedef name="aa-type">
    <description>
        <text>
            test type
        </text>
    </description>
    <type name="int32">
        <range value="1..10 | 20..30" />
    </type>
</typedef>
<container name="a">
    <leaf-list name="aa">
        <type name="aa-type" />
        <max-elements value="5" />
    </leaf-list>
</container>
<leaf name="b">
    <type name="leafref">
        <path value="../a/aa" />
    </type>
</leaf>
<list name="c">
    <key value="x" />
    <leaf name="x">
        <type name="uint16" />
    </leaf>
    <leaf name="y">
        <type name="instance-identifier" />
    </leaf>
</list>
</module>
```

## 5.3.2 HTML TRANSLATION

xHTML 1.0 files can be generated by using the **--format=html** parameter value.

An HTML version of a YANG file will contain color-coded YANG tokens to make the file easier to read. There will also be links created whenever possible, for statements which reference typedefs, groupings, extensions, etc.

The following configuration parameters are available to control some of the details:

- html-div**: If 'true', create a single <div> element that can be integrated into other WEB pages. The default is 'false', to create an entire WEB page (<html> element).
- html-toc**: controls how (or if) a table of contents section is generated. The default is to create a drop-down menu type ToC, which requires that Javascript is enabled in the browser.
- objview**: controls whether raw (with uses, augments, refine, and deviation statements) or cooked (final object tree without uses, etc). The default is the 'raw' (original) view.
- simurls**: controls how URLs with parameter fragments are generated. The default is 'false', which is to use traditional encoded parameters.
- unified**: controls whether submodules are combined into the main module, or if the 'include' statements are left in place. The default is 'false', to treat each file separately, and not expand any include statements.
- urlstart**: specifies the string that starts every generated URL in all A and HREF attributes. There is no default for this parameter.

The following example shows the HTML that is generated for --module=test4, using default values for all HTML generation parameters:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
  "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html lang="en" xmlns="http://www.w3.org/1999/xhtml">
<head>
  <title>YANG Module test4 Version 2009-09-09</title>
  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
  <meta name="description" content="YANG data model documentation"/>
  <meta name="generator" content="yangdump 0.9.7.473 (http://www.netconfcentral.org/)"/>
  <link rel="stylesheet" href="http://netconfcentral.org/static/css/style.css"
type="text/css"/>
</head>
<body>
  <h1 class="yang">test4.yang</h1>

  <ul id="nav">
    <li><a href="#">Typedefs</a>
      <ul>
        <li><a href="#aa-type.10">aa-type</a></li>
      </ul>
    </li>
  </ul>
```

# Yuma Tools User Manual

```
<li><a href="#">Objects</a>
  <ul>
    <li class="daddy"><a href="#a.15">a</a>
      <ul>
        <li><a href="#aa.16">aa</a></li>
      </ul>
    </li>
    <li><a href="#b.22">b</a></li>
    <li class="daddy"><a href="#c.26">c</a>
      <ul>
        <li><a href="#x.28">x</a></li>
        <li><a href="#y.29">y</a></li>
      </ul>
    </li>
  </ul>
</li>
</ul>

<br />
<div class="yang">
<pre>

<span class="yang_kw">module</span> <span class="yang_id">test4</span> {

<span class="yang_kw">yang-version</span> <span class="yang_str">1</span>;

<span class="yang_kw">namespace</span> <span
class="yang_str">"http://netconfcentral.com/ns/test4"</span>;

<span class="yang_kw">prefix</span> <span class="yang_str">"t4"</span>;

<span class="yang_kw">revision</span> <span class="yang_str">"2009-09-09"</span> {
  <span class="yang_kw">description</span> <span class="yang_str">"Initial revision."
</span>;
}

<a name="aa-type.10"></a><span class="yang_kw">typedef</span> <span class="yang_id">aa-
type</span> {
  <span class="yang_kw">type</span> <span class="yang_id">int32</span> {
    <span class="yang_kw">range</span> <span class="yang_str">"1..10 | 20..30"</span>;
  }
  <span class="yang_kw">description</span> <span class="yang_str">"test type"</span>;
}
```

## Yuma Tools User Manual

```
}

<a name="a.15"></a><span class="yang_kw">container</span> <span class="yang_id">a</span>
{
    <span class="yang_kw">config</span> <span class="yang_str">"true"</span>;
    <a name="aa.16"></a><span class="yang_kw">leaf-list</span> <span
class="yang_id">aa</span> {
        <span class="yang_kw">type</span> <span class="yang_id"><a href="#aa-type.17">aa-
type</a></span>;
        <span class="yang_kw">max-elements</span> <span class="yang_str">"5"</span>;
        <span class="yang_kw">ordered-by</span> <span class="yang_str">"system"</span>;
    }
} <span class="yang_cmt">// container a</span>

<a name="b.22"></a><span class="yang_kw">leaf</span> <span class="yang_id">b</span> {
    <span class="yang_kw">type</span> <span class="yang_id">leafref</span> {
        <span class="yang_kw">path</span> <span class="yang_str">"../a/aa"</span>;
    }
    <span class="yang_kw">config</span> <span class="yang_str">"true"</span>;
}

<a name="c.26"></a><span class="yang_kw">list</span> <span class="yang_id">c</span> {
    <span class="yang_kw">key</span> "<a href="#x.28">x</a>";
    <span class="yang_kw">config</span> <span class="yang_str">"true"</span>;
    <span class="yang_kw">ordered-by</span> <span class="yang_str">"system"</span>;
    <a name="x.28"></a><span class="yang_kw">leaf</span> <span class="yang_id">x</span> {
        <span class="yang_kw">type</span> <span class="yang_id">uint16</span>;
    }
}

<a name="y.29"></a><span class="yang_kw">leaf</span> <span class="yang_id">y</span> {
    <span class="yang_kw">type</span> <span class="yang_id">instance-
identifier</span>;
}
<span class="yang_cmt">// list c</span>
} <span class="yang_cmt">// module test4</span>

</pre>
</div>
</body>
</html>
```

The following picture shows how this WEB page looks in Firefox 3.0 on Ubuntu:

**test4.yang**

TypeDefs	Objects
----------	---------

```

module test4 {

    yang-version 1;

    namespace "http://netconfcentral.com/ns/test4";

    prefix "t4";

    revision "2009-09-09" {
        description "Initial revision.";
    }

    typedef aa-type {
        type int32 {
            range "1..10 | 20..30";
        }
        description "test type";
    }

    container a {
        config "true";
        leaf-list aa {
            type aa-type;
            max-elements "5";
            ordered-by "system";
        }
    } // container a

    leaf b {
        type leafref {
            path "../a/aa";
        }
        config "true";
    }

    list c {
        key "x";
        config "true";
        ordered-by "system";
        leaf x {
            type uint16;
        }

        leaf y {
            type instance-identifier;
        }
    } // list c
} // module test4

```

**5.3.3 XSD TRANSLATION**

XSD 1.0 files can be generated by using the **--format=xsd** parameter value.

XSD format may be deprecated in a future release because the official YANG translation format is DSDL. The XSD translation of a YANG file will contain an XSD representation of the cooked objects in the module. The top-level typedefs and objects will be translated. Local typedefs and groupings, extensions, identities, and other YANG constructs are not translated at this time.

The **--xsd-schemaloc** parameter can be used to specify the *schemaLocation* attribute value for the XSD.

## Yuma Tools User Manual

The data structures generated during XSD translation will be extensions to the NETCONF XSD defined in RFC 4741.

- A YANG **rpc** statement will be translated to an `<rpcOperationType>` element.
- A YANG **notification** statement will be translated to an `<eventType>` element.
- A YANG **leaf** and **leaf-list** statements will be translated to a `<simpleType>`, and corresponding `<element>` definition.
- A YANG **container** or **list** statement will be translated to a `<complexType>`, and corresponding `<element>` definition.
- A YANG choice statement will be translated to a `<choice>` element.
- A YANG top-level **typedef** statement will be translated to a `<simpleType>` element.
- A YANG **range**-statement will be translated to `<restriction>` elements or a union of `<restriction>` elements if the YANG range is non-contiguous.
- A YANG **pattern** statement will be translated into a `<pattern>` element.
- Multiple YANG **patterns** within a single typedef will be translated into nested inline `<simpleType>` elements, to preserve the AND logic, instead of the XSD OR logic.
- YANG **description** and **reference** statements are translated to `<documentation>` elements within an `<annotation>` element.
- YANG **header** constructs are translated into `<annotation>` elements that are defined in **yuma-ncx.yang**.

The following example shows the translation of test/test4.yang. The command options used are **--format=xsd** and **--module=test4**.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns="http://netconfcentral.com/ns/test4"
  targetNamespace="http://netconfcentral.com/ns/test4"
  elementFormDefault="qualified" attributeFormDefault="unqualified"
  xml:lang="en" version="2009-09-09"
  xmlns:ncx="http://netconfcentral.com/ncx"
  xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0">
  <xs:annotation>
    <xs:documentation>
      Converted from YANG file 'test4.yang' by yangdump version 0.9.7.471
    </xs:documentation>
    Module: test4
    Version: 2009-09-09
  </xs:annotation>
  <xs:appinfo>
    <ncx:source>
      /home/andy/swdev/yuma/trunk/netconf/modules/test/test4.yang
    </ncx:source>
  </xs:appinfo>
```

# Yuma Tools User Manual

```
<xs:appinfo>
  <ncx:revision>
    <ncx:version>2009-09-09</ncx:version>
    <ncx:description>
      Initial revision.
    </ncx:description>
  </ncx:revision>
</xs:appinfo>
</xs:annotation>

<xs:simpleType name="aa-type">
  <xs:annotation>
    <xs:documentation>test type
    </xs:documentation>
  </xs:annotation>
  <xs:union>
    <xs:simpleType>
      <xs:restriction base="xs:int">
        <xs:minInclusive value="1"/>
        <xs:maxInclusive value="10"/>
      </xs:restriction>
    </xs:simpleType>
    <xs:simpleType>
      <xs:restriction base="xs:int">
        <xs:minInclusive value="20"/>
        <xs:maxInclusive value="30"/>
      </xs:restriction>
    </xs:simpleType>
  </xs:union>
</xs:simpleType>

<xs:element name="a">
  <xs:annotation>
    <xs:appinfo>
      <ncx:config>true</ncx:config>
    </xs:appinfo>
  </xs:annotation>
  <xs:complexType>
    <xs:sequence>
      <xs:element name="aa" type="aa-type" minOccurs="0"
        maxOccurs="5">
        <xs:annotation>
          <xs:appinfo>
```

## Yuma Tools User Manual

```
<ncx:ordered-by>system </ncx:ordered-by>
</xs:appinfo>
</xs:annotation>
</xs:element>
<xs:element name="___.a.A___" minOccurs="0"
           maxOccurs="unbounded" abstract="true" />
</xs:sequence>
</xs:complexType>
</xs:element>

<xs:element name="b" type="xs:string" minOccurs="0">
<xs:annotation>
<xs:appinfo>
<ncx:config>true</ncx:config>
</xs:appinfo>
</xs:annotation>
</xs:element>

<xs:element name="c" minOccurs="0" maxOccurs="unbounded">
<xs:annotation>
<xs:appinfo>
<ncx:config>true</ncx:config>
<ncx:ordered-by>system</ncx:ordered-by>
</xs:appinfo>
</xs:annotation>
<xs:complexType>
<xs:sequence>
<xs:element name="x" type="xs:unsignedShort" />
<xs:element name="y" type="xs:string" minOccurs="0" />
<xs:element name="___.c.A___" minOccurs="0"
           maxOccurs="unbounded" abstract="true" />
</xs:sequence>
</xs:complexType>
<xs:key name="c_Key" >
<xs:selector xpath=". " />
<xs:field xpath="x" />
</xs:key>
</xs:element>

</xs:schema>
```

### 5.3.4 SQL TRANSLATION

SQL translation is reserved for future use. The **--format=sql** enumeration is not supported yet.

### 5.3.5 SQL Documentation Database Translation

SQL documentation database translation for the Netconf Central documentation database can be generated using the **--format=sqldb** parameter value. This option is only available in the SDK version of Yuma. Refer to the Yuma Developer Guide for more details.

### 5.3.6 Canonical YANG Translation

Canonical YANG files can be generated by using the **--format=yang** parameter value.

In this translation mode, all YANG constructs are generated in the same order, with the same indentation.

The order used is the order defined in the YANG specification ABNF. The order is somewhat arbitrary, but the purpose of this translation mode is to generate consistent documentation.

Normally, the order of top-level statements within a module is preserved in the canonical YANG translation. Only the sub-statements within the top-level statements may be reordered. Child objects are never reordered, just the sub-statements within them.

Range statements will be converted to canonical form. If there are contiguous range parts, then they will be combined.

Example range statement:

```
range "1 .. 10 | 11 | 12 .. 20";
```

Canonical form for this example:

```
range "1 .. 20";
```

If the **--unified** parameter is set to 'true', then all statements of a similar type will be grouped together, from all sub-modules included by the main module.

The **ncx:hidden** extension will cause objects containing this extension to be omitted during translation.

Comments are not preserved at this time. This will be addressed in a future release.

### 5.3.7 Copy and Rename YANG Files

Exact copies of a YANG file can be copied to another directory and renamed to the default naming scheme, using the **--format=copy** parameter value.

Only YANG files that have no errors will be copied. If the validation phase produces a non-zero error count, then the file will not be copied and renamed.

The **--defnames** parameter is set to 'true' in this mode. The default YANG file names will be created for each input file.

Example file *foo.yang*:

```
module foo {  
    namespace http://example.com/ns/foo;  
    prefix foo;  
    // header skipped  
    revision 2009-02-03 {  
        description "Initial version";
```

```
}
```

```
}
```

Example renamed file, if **--output=~/workdir:**

- If **--versionnames=true:**
  - ~/workdir/foo.2009-02-03.yang
- If **--versionnames=false:**
  - ~/workdir/foo.yang

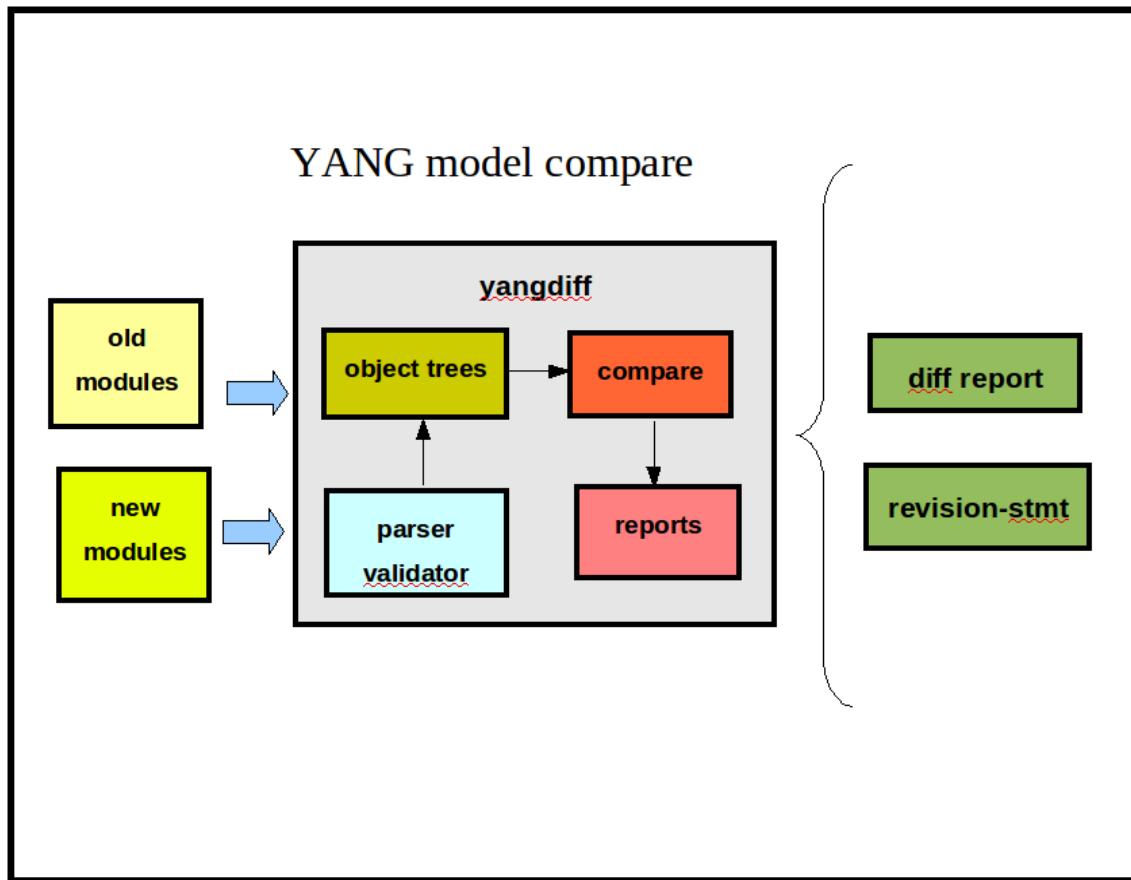
### 5.3.8 AGENT INSTRUMENTATION H FILE GENERATION

The **--format=h** parameter value is only available in the SDK version of Yuma. Refer to the Yuma Developer Guide for more details.

### 5.3.9 AGENT INSTRUMENTATION C FILE GENERATION

The **--format=c** parameter value is only available in the SDK version of Yuma. Refer to the Yuma Developer Guide for more details.

## 6 yangdiff User Guide



### 6.1 Introduction

The **yangdiff** program is used to compare two revisions of the same YANG file.

#### 6.1.1 FEATURES

The **yangdiff** program has the following features:

- The conceptual YANG object trees are compared, not the actual YANG statements.
- Two separate subtrees of modules can be compared, not just 1 file at a time.
- The differences report format for differences is easy to read, and can be configured with 2 different levels of verbosity.
- A YANG revision statement can be automatically generated instead of a list of differences. This will include an English text summary of the differences found.

## 6.1.2 STARTING YANGDIFF

The current working directory in use when **yangdiff** is invoked is important. It is most convenient to run **yangdiff** from a work directory, rather than the installation directory or within the module library.

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

- To get the current version and exit:

```
yangdiff --version
```

- To get program help and exit:

```
yangdiff --help
yangdiff --help --brief
yangdiff --help --full
```

- To compare a new YANG module named 'foo', with an old version named foo.2008-09-01.yang, the following command line can be used

```
yangdiff --old=foo.2008-09-01.yang --new=foo
```

- To get all the configuration parameters from a text file named '~/yangdiff-project1.conf':

```
yangdiff --config=~/yangdiff-project1.conf
```

- To generate a terse differences report called *~/project-X-diffs.log* for the old files in the '*/public/project-X/modules*' with the new modules in the '*~/work*' directory:

```
yangdiff --difftype=terse --output=~/project-X-diffs.log \
--old=/public/project-X/modules \
--new=/work
```

- To generate a revision statement differences to *~/foo-projectX-revision.txt* for the '*~/work/foo.yang*' module, with the old version in the in the '*/public/project-X/modules*':

```
yangdiff --difftype=revision --output=~/foo-project-X-revision.txt \
--old=/public/project-X/modules \
--new=/work/foo.yang
```

## 6.1.3 STOPPING YANGDIFF

There is no interactive mode for **yangdiff**, so there is no need for a command to exit the program.

The Control C character sequence can be used to cancel the **yangdiff** processing in progress. However, this will leave any partially completed output files in place.

## 6.1.4 CONFIGURATION PARAMETER LIST

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

### yangdiff CLI Parameters

parameter	description
--config	Specifies the configuration file to use for parameters.
--datapath	Sets the data file search path.
--difftype	Specifies the type of differences report that should be output.
--header	Specifies whether the module header data should be compared or not.
--help	Get context-sensitive help, then exit.
--help-mode	Adjust the help output (--brief, or --full).
--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.
--modpath	Sets the module search path.
--new	Specifies the location of the new revision to compare.
--old	Specifies the location of the old revision to compare.
--output	Specifies where output files should be generated.
--runpath	Sets the executable file search path.
--subdirs	Controls whether sub-directories are searched for YANG files.
--version	Prints the program version and then exit.
--warn-idlen	Controls how identifier lengths are checked.
--warn-linelen	Controls how line lengths are checked.
--warn-off	Suppresses the specified warning number.
--yuma-home	Specifies the <b>\$YUMA_HOME</b> project root to use when searching for files.

## 6.2 Comparing YANG Modules

The **yangdiff** program compares YANG files in the following manner:

- The cooked object trees are compared, not the actual YANG statements.
- Whitespace differences are ignored.
- Non-semantic changes are ignored:

## Yuma Tools User Manual

- A uses-stmt can replace a set of objects if the grouping is identical to the old objects.
- Refine statements are combined into the cooked objects, and not compared directly.
- A typedef statement can replace an inline type statement
- Changes to YANG comments are ignored.
- Sub-statement order is ignored, however object order with a container or list is not ignored.
- Objects can be moved to submodules, and include statements instead.

If the **--old** parameter is missing, then the module search path will be used to find the specified module with the same name. If the **--old** parameter contains just a module name, then the module search path will be used to find a module with the new name.

The **--new** parameter is required. It can represent one YANG file or a directory of new YANG modules.

The **--difftype** parameter is optional. The 'normal' report mode is used if this parameter is missing.

The **--output** parameter will be used for the report file, if it is specified.

symbol	description
<b>A</b>	Definition has been added.
<b>M</b>	Definition has been modified.
<b>D</b>	Definition has been deleted.

## 6.3 Diff Reports

This section uses the example module below (`test/test3a.yang`) to demonstrate the different report formats available. The old module revision is `test/test3.yang`.

The following command is used in all 3 examples, except the value of the **--difftype** parameter is changed each time.

```
yangdiff --old=test3a --new=test3 --difftype=<enum>
```

### 6.3.1 TERSE REPORT

If **--difftype=terse** is selected, then a brief summary of all changes will be listed. There will be no indentation, and only the change (Add, Modify, Delete), and the top-level definition is identified.

```
// Generated by yangdiff 0.9.7.473
// Copyright (c) 2009, Netconf Central, All Rights Reserved.

// old: test3 (2008-10-19) test3.yang
// new: test3 (2009-09-09) test3a.yang

D revision '2008-10-19'
```

```
A revision '2009-09-09'  
A feature X  
A identity testbase  
A identity testbasel  
M typedef C  
D container test-D1  
D leaf test-D  
M container test33
```

### 6.3.2 NORMAL REPORT

If **--difftype=normal** is selected, then a complete summary of all changes will be listed.

If a change line is indented, it indicates a sub-statement of the object in the previous line has been changed.

```
// Generated by yangdiff 0.9.7.473  
// Copyright (c) 2009, Netconf Central, All Rights Reserved.  
  
// old: test3 (2008-10-19) test3.yang  
// new: test3 (2009-09-09) test3a.yang  
  
D revision '2008-10-19'  
A revision '2009-09-09'  
A feature X  
A identity testbase  
A identity testbasel  
M typedef C  
    M type  
        M range from 'min .. 41 | 45' to 'min .. 41'  
D container test-D1  
D leaf test-D  
M container test33  
    D presence 'not a top-level mand...'  
    M choice f  
        M case f1  
            M leaf f1  
                A if-feature 'X'
```

### 6.3.3 REVISION STATEMENT

## Yuma Tools User Manual

If **--diff-type=revision** is selected, then a complete summary of all changes will be printed in the form of a YANG revision statement. The current date will be used for the revision-date field of the revision statement.

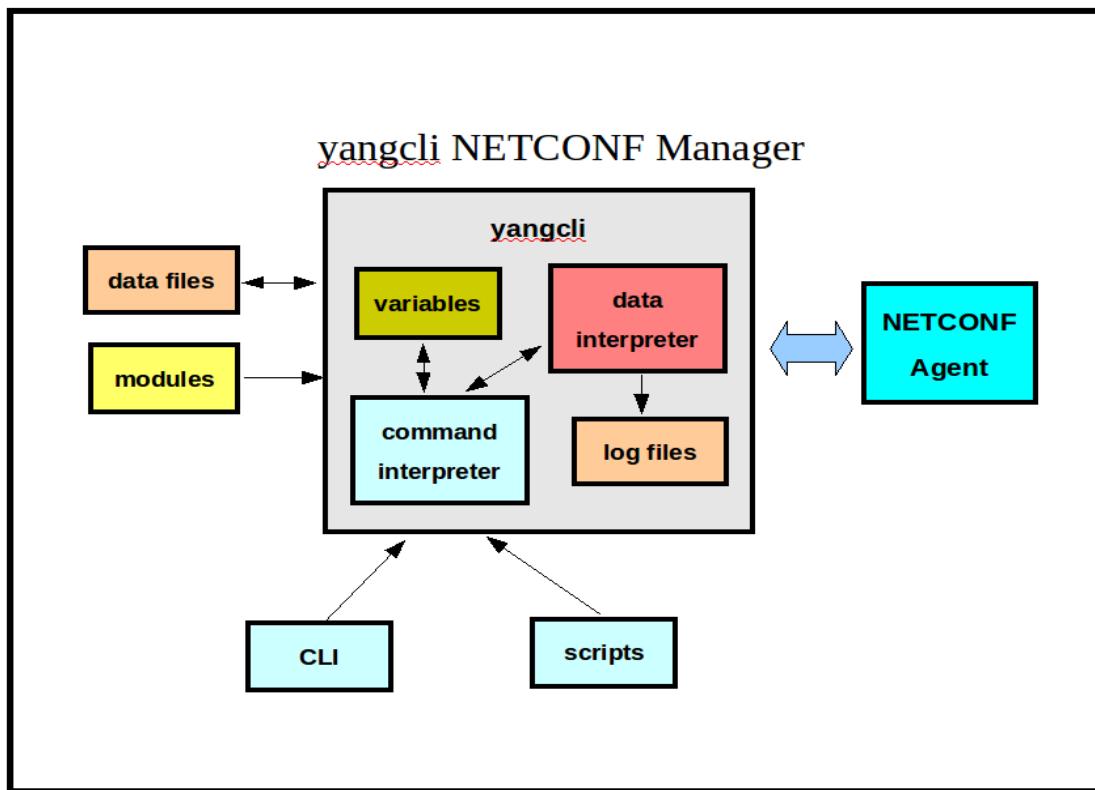
```
// Generated by yangdiff 0.9.7.473
// Copyright (c) 2009, Netconf Central, All Rights Reserved.

// old: test3 (2008-10-19) test3.yang
// new: test3 (2009-09-09) test3a.yang

revision 2009-09-10 {
    description "
        - Removed revision '2008-10-19'
        - Added revision '2009-09-09'
        - Added feature X
        - Added identity testbase
        - Added identity testbasel
        - Changed typedef C
            - Changed type
                - Changed range from 'min .. 41 | 45' to 'min .. 41'
        - Removed container test-D1
        - Removed leaf test-D
        - Changed container test33
            - Removed presence 'not a top-level mand...'
            - Changed choice f
                - Changed case f1
                - Changed leaf f1
                    - Added if-feature 'X'
    ";
}
```

# 7 yangcli User Guide

## Program Components



## 7.1 Introduction

The **yangcli** program is a NETCONF over SSH client application. It is driven directly by YANG modules, and provides a simple but powerful application interface for any YANG file. There is no configuration required at all to use any YANG file, although there are some YANG extensions that will be utilized if present.

### 7.1.1 FEATURES

The **yangcli** client has the following features:

- Automatic support for all NETCONF protocol operations, including several 'short-hand' commands for the most common operations, like <edit-config> and <get-config>.
- Automated database locking, unlocking and error cleanup, using the high-level **get-locks** and **release-locks** commands.
- Automatic, standards-based, server schema synchronization, using the YANG module capability URI information in the <hello> PDU, and the <get-schema> operation:

## Yuma Tools User Manual

- For each session, the exact view of the server schema definition tree is created, based on the module capability:
  - module namespace
  - module name
  - module revision date
  - enabled features
  - names of any modules that contain deviations for this module
- The help text and parameter validation for each session will be tailored to the capabilities advertised by the server.
- Parses each potential matching YANG file to make sure the module name, revision date, and namespace URI value are all exactly the same as the values in the module capability URI.
- Understands all NETCONF protocol capabilities, and complex hard-wired logic simplifies protocol usage, and allows high-level commands to pick appropriate defaults for many RPC operation parameters.
- Load any YANG module at boot-time or run-time and start using it immediately.
- Full concurrent support for multiple revisions of the same module.
- Supports NETCONF notifications, including :interleave capability.
- Full XPath 1.0 and subtree filtering support.
- Automatic support for all YANG language mechanisms, including extensions.
- Any YANG <rpc> operation is automatically available as a **yangcli** command.
- Uses YANG files directly as input, so no pre-processing or configuration needed to use a new module.
- Can be called from **unix** scripts in 'batch-mode' to automatically establish a NETCONF session, issue a command or invoke a yangcli script, close the session, and return the results.
- Extensive interactive shell environment, including user variables, file variables, smart parameter set completion, and a simple scripting environment for automatic operation.
- Automatic, context-sensitive (tab key) command-line completion.
- Full support for XPath, instance-identifier, leafref, and identityref parameters.
- Automatic, context-sensitive help system, based completely on YANG files and using the exact modules supported by the current NETCONF session, if connected.
- Full, customizable command line editing, using **emacs** by default, but **vi** or a custom set of keystroke bindings are also supported.
- Command line history and command recall.
- Store and recall command line history files for later use.
- Automatic NETCONF session management, including support for all YANG extensions to the <capability> element.
- Automatic recognition and support for all NETCONF 'capability' related operations.
- Automatic support for all YANG additions to the NETCONF protocol, such as the **insert** operation
- Unlimited nested scripts with up to 10 parameters each can automate testing and other management tasks

## 7.1.2 STARTING YANGCLI

The current working directory in use when **yangcli** is invoked is important. It is most convenient to run **yangcli** from a work directory, rather than the installation directory or within the module library.

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

- To get the current version and exit:

```
yangcli --version
```

- To get program help and exit:

```
yangcli --help
yangcli --help --brief
yangcli --help --full
```

- To start an interactive session with the default parameters:

```
yangcli
```

- To start an interactive session with a new log file:

```
yangcli --logfile=mylogfile
```

- To start an interactive session and append to an existing log file:

```
yangcli --logfile=mylogfile --log-append
```

- To get parameters from a configuration file:

```
yangcli --config=~/yangcli.conf
```

- To begin to connect to a server upon startup, provide the **--server** parameter. The **connect** command will be started upon startup and the user will be prompted to enter the rest of the mandatory parameters to the **connect** command.

```
yangcli server=myagent.example.com
```

- To connect to a server and automatically connect without any interactive interruption, enter the **--server**, **--user**, and **--password** parameters. A session startup will be attempted right away, using these parameters. Any optional parameters for the **connect** command (**--port** or **--timeout**) may be entered as well. All parameters can be entered from a config file, and/or the command line.

```
yangcli --server=myagent.example.com \
```

## Yuma Tools User Manual

```
--user=andy --password=yangrocks
```

- To automatically connect to a server, run a script in non-interactive mode, and then remain connected to the server, add the **--run-script** parameter to the connection parameters. The **--runpath** parameter can also be entered, if needed.

```
yangcli --server=myagent.example.com \
--user=andy --password=yangrocks \
--run-script=mytestscript
```

- To automatically connect to a server, run a script in non-interactive mode, and then exit the program, add the **--batch-mode** and **--run-script** parameters to the connection parameters. The **--runpath** parameter can also be entered, if needed.

```
yangcli --server=myagent.example.com \
--user=andy --password=yangrocks \
--run-script=mytestscript --batch-mode
```

- To automatically connect to a server, and run a single command instead of a script, and then exit, use the **--run-command** parameter instead of the **--run-script** parameter. The **--batch-mode** parameter can be left out to remain in the current session (in interactive mode) after the command is invoked.

```
yangcli --server=myagent.example.com \
--user=andy --password=yangrocks \
--batch-mode --run-command="sget /system"
```

### 7.1.3 STOPPING YANGCLI

To terminate the **yangcli** program, use the **quit** command.

The control-C character sequence will also cause the program to be terminated.

### 7.1.4 STATEMENTS

The **yangcli** script interpreter accepts several types of statements:

#### yangcli Statements

type	description	example
command	invoke a local command and/or send an <rpc> to the server	sget /system
variable assignment	set a user variable to some value	\$system = sget /system

## Yuma Tools User Manual

file assignment	set the contents of a file to some value	@save.txt = \$system
variable deletion	delete a user variable or clear a system variable	\$system =

A command can be as simple like 'get' or complex, like 'edit-config'.

A variable assignment sets the specified user or system variable to the right hand side of the expression. An expression has many forms, including the result from a local command or a remote NETCONF operation.

If a string that matches a command is used as the assignment value, then it must be entered in quotes (single or double). For example, the `$$default-operation` system configuration variable accepts enumeration values which also match RPC commands:

```
yangcli> $$display-mode = 'merge'
```

A file assignment is essentially the same as a variable assignment, except the right hand side of the expression is saved in the specified file, instead of a user variable.

To delete a user variable, leave the right hand side of the expression empty (or just whitespace).

Note: More statement types will be available in the next release.

### 7.1.5 COMMANDS

The yangcli program has several built-in commands, defined in **yangcli.yang**, **yuma-netconf.yang**, and **notifications.yang**.

The YANG **rpc** statement is used to define the syntax and behavior of each command.

There are 2 types of yangcli commands:

- **local**: the command is executed within the yangcli application, and can be invoked at any time.
- **remote**: the command is executed on the remote server, and is only available when a NETCONF session is active. Any YANG **rpc** statement that **yangcli** does not recognize as a local command is treated as a remote command available on the server.

#### Local Commands

command	description
cd	change the current working directory
connect	connect to a server and start a NETCONF session
eventlog	view or clear the notification event log
fill	fill a user variable
help	get context-sensitive help
history	manage the command history buffer
list	list modules, objects, or other properties of the session
mgrload	load a YANG file into the client only
pwd	print the current working directoty

## Yuma Tools User Manual

quit	exit the program
recall	recall a line from the command history buffer
run	run a script
show	show variables and objects currently available

The following table shows the standard NETCONF protocol operations that are directly available for use, depending on the capabilities of the server.

### Standard NETCONF Commands

command	description
close-session	Close the current NETCONF session
commit	Make the candidate database be the running config
copy-config	Copy an entire NETCONF database
create-subscription	Start receiving NETCONF notifications
delete-config	Delete an entire NETCONF database
discard-changes	Discard any edits in the candidate database
edit-config	Alteration of the target database
get	Filtered retrieval of state data and running config
get-config	Filtered retrieval of any NETCONF database
get-schema	Get a data model definition file from the server
kill-session	Force close another NETCONF session
lock	Lock a NETCONF database that is currently unlocked
unlock	Unlock a NETCONF database that is currently locked
validate	Validate the contents of a NETCONF database

The following **yangcli** commands are available for simplified access to standard NETCONF operations

### Custom NETCONF Commands

command	description
create	Invoke an <edit-config> create operation
delete	Invoke an <edit-config> delete operation
get-locks	Lock all the databases with the <lock> operation
insert	Invoke an <edit-config> YANG insert operation
merge	Invoke an <edit-config> merge operation
release-locks	Unlock all the locked databases with the <unlock> operation
replace	Invoke an <edit-config> replace operation
save	Save the current edits on the server in NV-storage

sget	Invoke a <get> operation with a subtree filter
sget-config	Invoke a <get-config> operation with a subtree filter
xget	Invoke a <get> operation with an XPath filter
xget-config	Invoke a <get-config> operation with an XPath filter

The following table shows the extended NETCONF protocol operations that are available on the **netconfd** server only.

### **Extended netconfd Commands**

command	description
get-my-session	Retrieve session customization parameters
load	Load a module into the server.
no-op	No operation.
restart	Restart the server.
set-log-level	Set the server logging verbosity level.
set-my-session	Set session customization parameters.
shutdown	Shutdown the server.

## 7.1.6 VARIABLES

The **yangcli** program utilizes several types of user-accessible variables. These variables can be listed with the '**show vars**' command.

A variable reference consists of 1 or 2 dollar signs ('\$'), followed immediately by a valid identifier string (e.g., **\$\$global-log** or **\$local-log**).

Variables can be 1 or more characters in length, and follow the YANG rules for identifier names. The first character must be a letter, 'A' to 'Z', or 'a' to 'z'. The 2<sup>nd</sup> to last characters can be a letter 'A' to 'Z', or 'a' to 'z', number ('0' to '9'), an underscore ('\_'), a dash ('-'), or a period ('.') character.

There are 4 categories of parameters supported:

1. Read-only system variables
2. Read-write system variables
3. Read-write global user variables
4. Read-write local user variables

It is an error if a variable is referenced (in the right-hand-side of a statement) that does not exist.

The first 3 types are **global variables**, which means that they are available to all run-levels of all scripts. The last type, called a **local variable**, is only visible to the current run-level of the current script (or interactive shell). Refer to the following section for more details on run levels.

## Variable Syntax

<b>syntax</b>	<b>description</b>	<b>example</b>
\$\$<variable-name>	Left hand side: set the global variable to some value	\$\$saved_get = get
\$\$<variable-name>	Right hand side: access the value of a global variable	fill --target=\ \$\$mytarget
\$<variable-name>	Left hand side: set the local variable to some value	\$myloglevel = \ \$\$log-level
\$<variable-name>	Right hand side: access the value of any variable with this name (try local, then global)	\$myuser = \$user

The following table shows the **unix** environment variables that are available as read-only global variables in **yangcli**. These variables are set once when the program is started, and cannot be used in the left hand side of an assignment statement.

## Read-only system variables

<b>variable</b>	<b>description</b>
\$\$HOME	the <b>HOME</b> environment variable
\$\$HOSTNAME	the <b>HOST</b> or <b>HOSTNAME</b> environment variable
\$\$LANG	the <b>LANG</b> environment variable
\$\$PWD	the <b>PWD</b> environment variable, when <b>yangcli</b> was invoked
\$\$SHELL	the <b>SHELL</b> environment variable
\$\$USER	the <b>USER</b> environment variable
\$\$YUMA_DATAPATH	the <b>YUMA_DATAPATH</b> environment variable
\$\$YUMA_HOME	the <b>YUMA_HOME</b> environment variable
\$\$YUMA_MODPATH	the <b>YUMA_MODPATH</b> environment variable
\$\$YUMA_RUNPATH	the <b>YUMA_RUNPATH</b> environment variable

The following table shows the CLI configuration parameters that can be read or changed (but not deleted). If a particular parameter was not set during program invocation, then the associated variable will contain the empty string.

## Read-write system variables

<b>variable</b>	<b>description</b>
\$\$server	the <b>--server</b> configuration parameter
\$\$autocomp	the <b>--autocomp</b> configuration parameter
\$\$autoload	the <b>--autoload</b> configuration parameter

\$\$baddata	the <b>--baddata</b> configuration parameter
\$\$default-operation	the <default-operation> parameter for the <edit-config> operation
\$\$default-module	the <b>--default-module</b> configuration parameter
\$\$display-mode	the <b>--display-mode</b> configuration parameter
\$\$error-option	the <error-operation> parameter for the <edit-config> operation
\$\$fixorder	the <b>--fixorder</b> configuration parameter
\$\$log-level	the <b>--log-level</b> configuration parameter
\$\$optional	the <b>--optional</b> configuration parameter
\$\$test-option	the <test-operation> parameter for the <edit-config> operation
\$\$timeout	the <b>--timeout</b> configuration parameter
\$\$user	the <b>--user</b> configuration parameter
\$\$with-defaults	the <b>--with-defaults</b> configuration parameter

## Read-write global user variables

If a unrecognized global variable (e.g., \$\$foo) is used in the left-hand side of an assignment statement, then a global user variable will be created with that name. If the global user variable already exists, then its value will be overwritten.

## Read-write local user variables

If a local variable (e.g., \$foo) is used in the left-hand side of an assignment statement, then a local user variable will be created with that name. If the local user variable already exists, then its value will be overwritten. If the variable is created within a script (i.e., run-level greater than zero), then it will be deleted when the script exits.

## **7.1.7 FILES**

File contents can be used in **yangcli** statements, similar to user variables.

A file reference consist of the 'at-sign' character ('@') followed immediately by a valid file specification.

```
@foo.yang = get-schema --identifier=foo --version="" --format="YANG"

mgrload --module=foo
```

If the file extension is “.yang”, “.log”, “.txt”, or “.text”, then the value (or command output) will be saved, and yangcli will strip off the outermost XML (if needed) to save the requested file as a pure text file. Otherwise, the file will be saved in XML format.

Note: The **--display-mode** configuration parameter, and **\$\$display-mode** system variable, only affect the output of data in file assignment statements if the file name has one of these extensions. Otherwise, the output will be in XML format.

Files may also be used as parameter replacements, within a command.

```
$saved_get = get --filter=@filter.xml --with-defaults=explicit
```

It is an error if the file referenced does not exist or cannot be read.

### 7.1.8 SCRIPTS

Any command can be entered at the interactive shell, or stored in a script file, and invoked with the '**run**' command. Scripts are simply text files, and the extension used does not matter.

There are no formal templates for scripts, like there are for RPC operations, at this time. Instead, positional parameters can be passed to any script.

The parameters named **--P1** to **--P9** allow up to 9 parameters to be passed to any script. Within each script, the numbered parameters '**\$1**' to '**\$9**' are available, and contain the value that was passed as the corresponding **--Pn**" parameter when calling the script.

If a line contains only optional whitespace, followed by the pound sign character '#', then the line is treated as a comment. These lines will be skipped.

If an error occurs during a command within a script, or an <rpc-error> is received from the server during a NETCONF session, then the running script will be canceled, and if this was a nested script, then all parent scripts will also be canceled.

Script Example:

```
run connect --P1=andy --P2==localhost --P3=yangrocks

// connect script
# start a NETCONF session
$user = $1
$server = $2
$password = $3

connect --user=$user --server=$server --password=$password
```

### Run Levels

The **run** command can appear in a script.

When **yangcli** starts up, either in interactive mode or in batch mode, the script interpreter is at run level zero. Each time a **run** command is invoked, either at the command line or within a script currently being invoked, a new run level with the next higher value is assigned. Local variables are only visible within the current run level.

A maximum of 64 run levels are supported in **yangcli**.

## 7.1.9 CONFIGURATION PARAMETER LIST

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

### yangcli CLI Parameters

parameter	description
--autocomp	Controls whether partial commands are allowed or not.
--autohistory	Controls whether the command line history buffer will be automatically loaded at startup and saved on exit.
--autoload	Controls whether modules used by the server will be loaded automatically, as needed.
--bad-data	Controls how bad data about to be sent to the server is handled.
--batch-mode	Indicates the interactive shell should not be used.
--config	Specifies the configuration file to use for parameters.
--datapath	Sets the data file search path.
--default-module	Specifies the default module to use to resolve identifiers.
--default-operation	Specifies the enumeration to use for the <default-operation> parameter to the <edit-config> operation, if none is provided.
--deviation	Specifies one or more YANG modules to load as deviations.
--display-mode	Specifies how values should be displayed.
--fixorder	Controls whether PDUs are changed to canonical order before sending them to the server.
--help	Get program help.
--help-mode	Adjust the help output (--brief or --full).
--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.
--modpath	Sets the module search path.
--module	Specifies one or more YANG modules to load upon startup.
--password	Specifies the password to use in the <b>connect</b> command.
--port	Specifies the port number to use in the <b>connect</b> command.
--runpath	Sets the executable file search path.

--run-command	Specifies the command to run at startup time.
--run-script	Specifies the script to run at startup time.
--server	Specifies the server address to use in the <b>connect</b> command.
--timeout	Specifies the timeout to use in the <b>connect</b> command.
--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.
--yuma-home	Specifies the <b>\$YUMA_HOME</b> project root to use when searching for files.

## 7.2 Invoking Commands

Commands can be entered with parameters:

- in one continuous line  

```
merge target=/toaster/toastControl --value="down"
```
- in several lines (using line continuation)  

```
merge target=/toaster/toastControl \
--value="down"
```
- interactively prompted for each parameter  

```
merge
(will be prompted for target and value)
```
- a combination of the above  

```
merge target=/toaster/toastControl
(will be prompted for value)
```

When a command is entered, and the yangcli script interpreter is running in interactive mode (**--batch-mode** not active), then the user will be prompted for any missing mandatory parameters.

If the **--optional** parameter is present (or the **\$optional** system variable is set to 'true'), then the user will be prompted for any optional parameters that are missing.

A command has the basic form:

<name (QName)> <parameter (any YANG type)>\*

The command name is a qualified name matching the name used in the YANG rpc statement. This is followed by zero or more parameters (in any order).

A command parameter has the same syntax as a CLI configuration parameter.

The command name syntax is described below.

An un-escaped end-of-line character ('enter key') terminates command line input.

### 7.2.1 COMMAND PROMPT

## Yuma Tools User Manual

The **yangcli** command prompt changes, depending on the context.

### **Idle Mode:**

If the script interpreter is idle and there is no NETCONF session active, then the prompt is simply the program name:

```
yangcli>
```

If the script interpreter is idle and there is a NETCONF session active, then the prompt is the program name, followed by '**<user>@<server>**', depending on the parameters used in the **connect** command:

```
yangcli andy@myagent>
```

### **Continuation Mode:**

If a backslash, end-of-line sequence ended the previous line, the prompt will simply be the word 'more' indented 3 spaces:

```
yangcli andy@myagent> get \
more>
```

The 'more>' prompt will continue if the new line also ends in with an escaped end-of-line. When a new line is finally terminated, all the fragments are spliced together and delivered as one command line.

Note: context-sensitive command completion is not available in this mode.

### **Choice mode:**

If a partial command has been entered in interactive mode, and the script interpreter needs to prompt for a YANG 'choice' parameter, then a list of numbered cases will be presented, and the prompt will be the same as it was before (depending on whether a NETCONF session is active or not), except a colon character (':'), followed by the command name, will be added at the end. As long as parameters for the same command are being entered (i.e., prompted for child nodes within a selected case, the command name will be appended to the prompt).

```
yangcli andy@myagent> sget
```

```
Enter a number of the selected case statement:
```

## Yuma Tools User Manual

```
1: case varref:  
    leaf varref  
2: case from-cli:  
    leaf target  
    leaf optional  
    anyxml value  
  
Enter choice number [1 - 2]:  
yangcli andy@myagent:sget>
```

### **Parameter mode:**

If a partial command has been entered in interactive mode, and the script interpreter needs to prompt for a leaf or leaf-list, then the parameter name will appear in angle brackets ('<' and '>').

```
Filling mandatory case /sget/input/from/from-cli:  
Enter string value for leaf <target>  
yangcli andy@myagent:sget>
```

If the '**ncx:password**' extension is part of the YANG definition for the leaf or leaf-list, then the characters entered at the prompt in this mode will not be echoed, and they will not be saved in the command history buffer. Any default value will not be printed either. Instead, 4 asterisks '\*\*\*\*' will be printed, even though the correct value will be used in the command.

If a default value is available, it will appear in square brackets ('[' and ']'). In this case, entering 'return' will not be interpreted as an empty string, but rather the default value that was presented.

```
yangcli> connect  
  
Enter string value for leaf <user> [andy]  
yangcli:connect>  
  
Enter string value for leaf <server> [myagent]  
yangcli:connect>  
  
Enter string value for leaf <password> [****]  
yangcli:connect>  
Enter uint16 value for leaf <port> [830]  
  
yangcli:connect>  
  
Enter uint32 value for leaf <timeout> [30]
```

```
yangcli:connect>
```

Note: After a NETCONF session is terminated for any reason, the connection parameters will be remembered , and presented as defaults the next time the connect command is entered.

## 7.2.2 COMMAND NAME

The command name can be entered with or without an XML prefix:

```
yangcli andy@myagent> nc:get  
yangcli andy@myagent> get
```

If there is a prefix (e.g., 'nc:get'), then it needs to be one of the XML prefixes in use at the time. Use the 'show modules' command to see the modules and prefixes in use. The YANG prefix will usually be the same as the XML prefix, but not always.

XML prefixes are required to be unique, so if any 2 YANG modules pick the same prefix, then 1 of them has to be changed for XML encoding purposes.

If the **--default-module** configuration parameter (or **\$\$default-module** system variable) is set, it will be used to resolve a command name without any prefix, if it is not a NETCONF or **yangcli** command.

An error message will be printed if the command entered cannot be found in any YANG module, or if there are multiple commands that match the same string.

## 7.2.3 NCX:DEFAULT-PARM EXTENSION

Each command may define a default parameter, by placing an '**ncx:default-parm**' extension in the rpc input section in the YANG rpc statement. This extension allows less typing in **yangcli** to accomplish the same thing.

If the script interpreter encounters a string in the command line that is not a recognized parameter for that command, and there is a default parameter defined, then the string will be used as a value for the default parameter.

For example, the 'show' parameter is the default parameter for the 'history' command, so both of these commands will be interpreted to mean the same thing:

```
history --show=10  
history 10
```

Note: the default parameter does not allow the wrong form of a parameter type to be entered, to accept the default for that parameter. For example, the 'show' parameter above has a default value of '25':

```
# this is the same as history show=25  
history  
  
# this is an error, not the same as the above  
history show
```

## Yuma Tools User Manual

The following table shows the default parameters that are available at this time.

### **Default Parameters**

<b>command</b>	<b>default parameter</b>
cd	dir
connect	server
create	target
eventlog	show
fill	target
help	command
history	show
insert	target
load	module
merge	target
mgrload	module
recall	index
replace	target
run	script
set-log-level	log-level
sget	target
sget-config	target
xget	select
xget-config	select

#### **7.2.4 PARAMETER MODE ESCAPE COMMANDS**

There are 4 escape sequence commands that can be used while entering parameters.

They all begin with the question mark character ('?'), and end with the 'Enter' key.

Control key sequences are not used because that would interfere with command line editing keys.

#### **Parameter mode escape sequences**

<b>escape sequence</b>	<b>description</b>
?	Print some help text
??	Get all available help text
?s	Skip this parameter
?c	Cancel this command

Note: If the current parameter is considered hidden (**ncx:hidden** extension used in the YANG definition), then no help will be available for the parameter, even though it is accessible. This also applies to the **help** command. Any command or parameter designated as **ncx:hidden** will be treated as an unknown identifier, and no help will be given.

Note: Skipping mandatory nodes with the '?s' command is affected by the **--bad-data** configuration parameter and **\$\$bad-data** system variable. An error, warning, or confirmation check may occur. Refer to the CLI Reference for more details.

Note: If there are any YANG defined values (e.g., enumeration, bits, default-stmt) available for the current parameter, then pressing the tab key will list the full or partial completions available.

### 7.2.5 USING XPATH EXPRESSIONS

There are some command parameters, such as the **--target** parameter for the **create** command, that accept XPath absolute path expressions.

If prefixes are present, then they must match the set of XML prefixes in use at the time. Use the **show modules** command to see the current set of prefixes.

If prefixes are not used, then the first available matching XML namespace will be used instead.

If the starting forward slash character ('/') is missing, then it will be added.

```
# these are all the same value
yangcli:fill> system
yangcli:fill> /system
yangcli:fill> /sys:system
```

It is important to remember 2 simple rules to avoid common errors in XPath expressions:

1. String constants must be quoted with single quote characters.  
The expression [name=fred] is not the same as [foo='fred'].  
The former compares the 'name' node value to the 'fred' node value.  
The latter compares the 'name' node value to the string 'fred'.
2. The double quote character ("") is not allowed in XPath **--select** parameter expressions because the expression will be sent to the server inside a double-quoted string.

If an incomplete XPath absolute path expression is entered, and the script interpreter is in interactive mode, then the user will be prompted to fill in any missing mandatory nodes or key leafs.

```
# complete form of ifMtu leaf
yangcli:fill> /interfaces/interface[name='eth0']/ifMtu
# incomplete form of ifMtu leaf

yangcli:fill> /interfaces/interface/ifMtu

Filling key leaf <name>:
Enter string value:
```

The **--select** parameter for the **xget** and **xget-config** commands accepts full XPath expressions. The expression must yield a node-set result in order to be used as a filter in NETCONF **get** and **get-config** operations.

One of the simplest XPath filters to use is the **descendant-or-self** filter ('//<expr>').

For example, this command will retrieve all instances of the 'ifMtu' leaf:

```
xget //ifMtu
```

When interface (or any list) entries are returned by netconfd, they will contain the the entire path back to the root of the YANG module, not just the specified node. Also, any key leafs within a list are added. This is only done if the XPath expression is missing any predicates for key leafs.

This is different than XPath 1.0 as used in XSLT. Since NETCONF **get** and **get-config** operations return complete XML instance documents, not node-sets, the ancestor nodes and naming nodes need to be added.

```
# reply shown with --display-mode=plain
data {
    interfaces {
        interface eth0 {
            name eth0
            ifMtu 1500
        }
        interface eth1 {
            name eth1
            ifMtu 1518
        }
    }
}
```

### 7.2.6 SPECIAL PARAMETER HANDLING

Some special handling of YANG data structures is done by the script interpreter.

#### Containers

Some parameters, such as the **--source** and **--target** parameters in many commands, are actually represented as a container with a single child -- a choice of several leaf nodes. In this situation, just the name of the desired leaf node can be entered (when in idle mode), as the 'contents' of the container parameter.

```
sget-config /system source=candidate
```

#### Choices and Cases

If a parameter name exact-match is not found, and a partial match is attempted, then choice and case node names will be ignored, and not cause a match.

Since these nodes never appear in the XML PDUs they are treated as transparent nodes (wrt/ parameter searches) unless they are specified with their full name.

Parameters that are a choice of several nodes, similar to above, except without a parent container node, (e.g., **--help-mode**) can be omitted. The accessible child nodes within the case nodes can be entered directly (e.g., **sget --target** parameter).

```
# this is not allowed because 'help-mode' is not complete
yangcli> help --command=help --help-mo=brief

# this is allowed because 'help-mode' is complete,
# even though help-mode is a choice and 'brief' is
# an empty leaf
yangcli> help help help-mode=brief

# choice and case names are transparent when
# searching for parameter names, so the
# following command is the same as above
yangcli> help help brief
```

## Lists and Leaf-Lists

When filling a data structure and a descendant node is entered, which is a YANG list or leaf-list, then multiple entries can be entered. After the node is filled in, there will be a prompt (Y/N, default no) to add more list or leaf-list entries.

## Binary Data Type

The YANG binary data type is supported. Parameters of this type should be entered in plain text and they will be converted to binary format.

### **7.2.7 COMMAND COMPLETION**

The 'tab' key is used for context-sensitive command completion:

- If no command has been started, a list of available commands is printed
- If a partial command is entered, a list of commands which match the characters entered so far is printed
- If a command is entered, but no parameters, then a list of available parameters is printed
- If a command is entered, and the cursor is within a command name, then a list of parameters which match the characters entered so far is printed

## Yuma Tools User Manual

- If a command is entered, and the cursor is after a command name, but not within a value string, then a list of available parameters is printed
- If a command is entered, and the cursor is within a command value, then a list of possible values which match the characters entered so far is printed. Note that not all data types support value completion at this time.
- If no values are available, but a default value is known, that value will be printed

Command list example: no NETCONF session is active:

```
yangcli> <hit tab key>
cd      fill      history   mgrload  quit      run
connect  help      list       pwd       recall    show
```

Command list example: NETCONF session is active

```
yangcli andy@myagent.example.com> <hit tab key>
cd                  get-schema          recall
close-session       help                replace
commit              history             restart
connect             insert              run
copy-config         kill-session        save
create              list                sget
create-subscription load               sget-config
delete              load-config         show
delete-config       lock                shutdown
discard-changes    merge               unlock
edit-config         mgrload            validate
fill                no-op               xget
get                 pwd                xget-config
get-config          quit
```

### 7.2.8 COMMAND LINE EDITING

The command line parser is based on **libtecla**, a freely available library.

The home page is located here:

<http://www.astro.caltech.edu/~mcs/tecla/>

The complete user guide for configuring **libtecla** is located here:

<http://www.astro.caltech.edu/~mcs/tecla/tecla.html>

If the file **\$HOME/.teclarc** exists, then **libtecla** will use it to configure the key bindings.

## Yuma Tools User Manual

By default, **libtecla** uses **emacs** key bindings. There is no need for any further **libtecla** configuration if **emacs** mode is desired.

In order to use the **vi** editor key bindings, the **\$HOME/.teclarc** file must exist, and it must contain the following line:

```
edit-mode vi
```

Custom key bindings are also available. Refer to the **libtecla** documentation for more details on command line editing key customization.

The control key sequence (^F == control key and f key at the same time). The letter is not case-sensitive, so ^F and ^f are the same command.

The alt key sequence (M-f == alt key and f key at the same time). The letter is not case-sensitive, so M-F and M-f are the same command.

The following table shows the the most common default key bindings:

**Common editing key bindings**

command	description
^F	cursor right
^B	cursor-left
^A	beginning of line
^E	end of line
^U	delete line
M-f	forward-word
M-b	backward word
^P	up history
^N	down history

### 7.2.9 COMMAND HISTORY

Each command line is saved in the command history buffer, unless a password is being entered in parameter mode.

By default, the previous history line (if any) will be shown if the ^P key is pressed.

By default, the next history line (if any) will be shown if the ^N key is pressed.

In addition, the **history** command can be used to control the command line buffer further. This command has 4 sub-modes:

- **show**: show maximum of N history entries (default is 25)
- **clear**: clear the history buffer
- **save**: save the history buffer to a file
- **load**: load the history buffer from a file

## Yuma Tools User Manual

By default, the command line history buffer is loaded when the program starts, and saved when the program exits. This behavior can be turned off by setting the **--autohistory** configuration parameter to 'false'.

Refer to the Command Reference section for more details on the **history** command.

### 7.2.10 COMMAND RESPONSES

The command output and debugging messages within yangcli is controlled by the current log level (error, warn, info, debug, debug2, debug3).

If a command is executed by the script interpreter, then a response will be printed, depending on the log level value.

If the log level is 'info' or higher, and there were no errors and no response, then the string 'OK' is printed.

```
yangcli> $foo = 7
          OK
yangcli>
```

If the log-level is set to 'error' or 'warn', then the 'OK' messages will be suppressed.

If the log level is set to 'debug' or higher, then responses from the server will be echoed to the log (file or STDOUT). The current display mode will be used when printing data structures such as <rpc-error> and <notification> element contents.

If an error response is received from the server, it will always be printed to the log.

```
yangcli andy@myagent> create /system

Filling container /system:
RPC Error Reply 5 for session 8:

rpc-reply {
    rpc-error {
        error-type application
        error-tag access-denied
        error-severity error
        error-app-tag limit-reached
        error-path /nc:rpc/nc:edit-config/nc:config/sys:system
        error-message 'max-access exceeded'
    }
}

yangcli andy@myagent>
```

Refer to the the **--log-level** parameter in the CLI Reference for more details.

## 7.3 NETCONF Sessions

The **yangcli** program can be connected to one NETCONF server at a time.

Run multiple instances of yangcli to control multiple agents at once.

Use the connect command to start a NETCONF session.

This section explains how to use **yangcli** to manage a NETCONF server, once a session is established.

When a NETCONF session starts up, a <capability> exchange is done, and the server reports exactly what content it supports. This information is used extensively to customize the session, and report errors and warnings for the session.

### 7.3.1 CONNECTION STARTUP SCREEN

If the **--log-level** is set to 'info' or higher, then a startup screen will be displayed when a NETCONF session is started. It contains:

- startup banner
- client session ID
- server session ID
- protocol capabilities supported by the server
  - Includes revision-date of supported module
- YANG modules supported by the server
  - Includes any features and deviations supported in the module
- Enterprise specific capabilities supported by the server
- Default target database (<candidate> or <running>)
- Save operation mapping for the server
- **with-defaults** reporting capability reported by the server

The following example shows a typical startup screen connecting to the **netconfd** server:

```
NETCONF session established for andy on myagent
```

```
Client Session Id: 1
Server Session Id: 8
```

```
Server Protocol Capabilities
Protocol Version: RFC 4741
candidate:1.0
rollback-on-error:1.0
validate:1.0
xpath:1.0
notification:1.0
interleave:1.0
```

## Yuma Tools User Manual

```
with-defaults:1.0
netconf-monitoring:1.0
schema-retrieval:1.0
```

### Server Module Capabilities

```
ietf-inet-types/2009-05-13
ietf-netconf-state/2009-03-03
ietf-with-defaults/2009-04-10
ietf-yang-types/2009-05-13
nacm/2009-05-13
nc-notifications/2008-07-14
ncx/2009-06-12
ncx-app-common/2009-04-10
ncxtypes/2008-07-20
netconfd/2009-05-28
notifications/2008-07-14
system/2009-06-04
test/2009-06-10
```

#### Features:

```
feature1
feature3
feature4
```

### Server Enterprise Capabilities

```
None
```

```
Default target set to: <candidate>
```

```
Save operation mapped to: commit
```

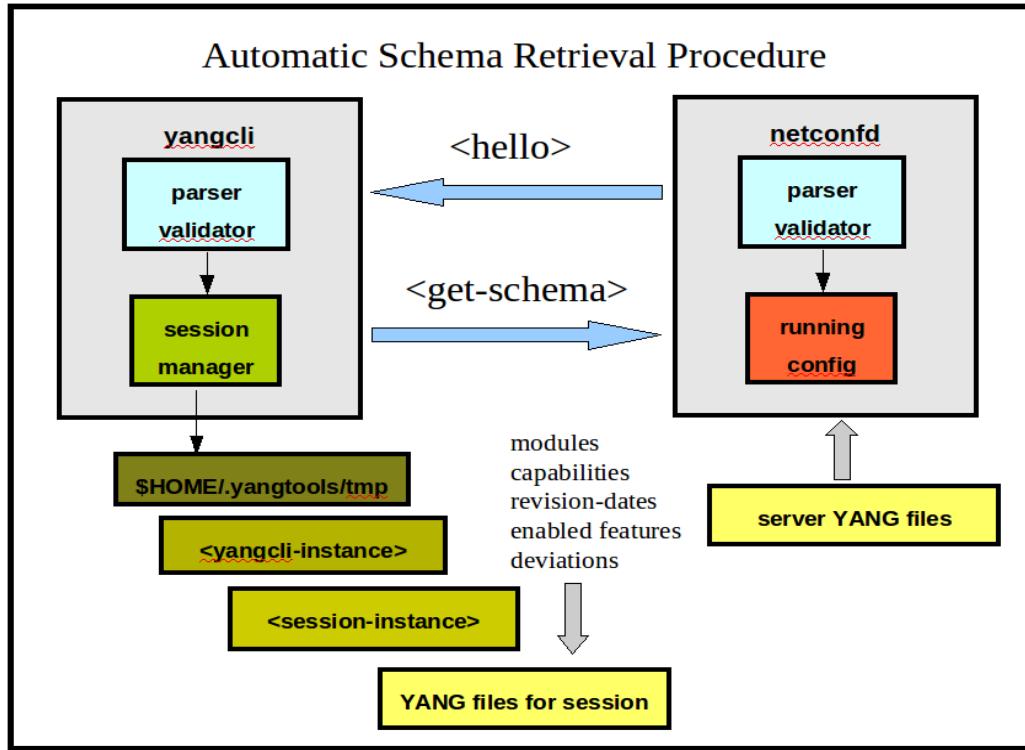
```
Default with-defaults behavior: explicit
```

```
Additional with-defaults behavior: trim:report-all
```

```
Checking server Modules...
```

```
yangcli andy@myserver>
```

### 7.3.2 SERVER TAILORED CONTEXT



While a NETCONF session is active, the set of available YANG modules will be set to the modules that the server is using, if the **--autoload** configuration parameter is enabled.

If the **:schema-retrieval** capability is also available on the server, then the **<get-schema>** operation will be attempted for any YANG module specified in the **<hello>** message capabilities, but not available to the **yangcli** program.

When the server module capabilities are analyzed by the **yangcli** client, the entire YANG module search path is checked for the specific module advertised in the capability. All the modules are partially parsed to check the actual namespace and revision date values. The following fields must exactly match in order for yangcli to use a local YANG module, if **--autoload=true**.

- module name
- module revision date (if any)
- module namespace

If the namespace URI value is different, it indicates that there is either a bug in one of the conflicting YANG modules, or that two different naming authorities picked the same module name. In this case, a warning will be generated during session initialization.

Any data returned from the server for YANG modules not currently available will be treated as a YANG 'anyxml' node, instead of the (unknown) YANG data type.

If the module contains YANG features that are not advertised in the **<capabilities>** exchange, then those data definitions will not be available (by default) for use in **yangcli** commands.

If the module contains an object with a 'when' statement, and the 'when' XPath expression evaluates to 'false', then that data definition will not be available (by default) for use in **yangcli** commands.

## Yuma Tools User Manual

The **help** command will be tailored to the modules, capabilities, features, and module deviations reported by the server in <capability> exchange.

### 7.3.3 RETRIEVING DATA

There are 6 commands available to retrieve generic data (i.e., an arbitrary subset of an entire NETCONF database):

command	description
get	raw NETCONF <get> operation
get-config	raw NETCONF <get-config> operation
sget	high-level subtree filter, using the <get> protocol operation
sget-config	high-level subtree filter, using the <get-config> protocol operation
xget	high-level XPath filter, using the <get> protocol operation
xget-config	high-level XPath filter, using the <get-config> protocol operation

All the high-level retrieval operations support the **\$\$with-defaults** system variable. The <with-defaults> parameter will be added to the NETCONF PDU if this variable is set to a value other than 'none' (the default). This system variable will be used as the default if not entered directly.

```
sget /system --with-defaults=$$with-defaults
```

This parameter can also be specified directly, each time the command is used.

```
xget-config //ifMtu --with-defaults=trim
```

The **\$\$bad-data** system variable is used to control how invalid operations and data are sent to the server. The **xget** and **xget-config** commands are affected by this parameter. If the :xpath capability was not advertised by the server when the session started, an error or warning may occur if these commands are used.

If any data is received that **yangcli** does not understand, then a warning message will be printed and the data will be treated as if it was defined with the YANG '**anyxml**' data type.

### 7.3.4 MODIFYING DATA

The following commands are available to modify generic data (i.e., an arbitrary subset of an entire NETCONF database):

command	description
commit	raw NETCONF <commit> operation

## Yuma Tools User Manual

create	high-level <edit-config> operation, with nc:operation='create'
delete	high-level <edit-config> operation, with nc:operation='delete'
delete-config	raw NETCONF <delete-config> operation
discard-changes	raw NETCONF <discard-changes> operation
edit-config	raw NETCONF <edit-config> operation
fill	fill a variable for re-use in other operations
insert	high-level <edit-config> operation, with YANG insert operation extensions
lock	lock a NETCONF database
merge	high-level <edit-config> operation, with nc:operation='merge'
replace	high-level <edit-config> operation, with nc:operation='replace'
save	High level save operation, depending on the default target (candidate or running)
unlock	unlock a NETCONF database

All the high-level editing operations use the **--target** parameter reported by the server when the session started up. If the server did not report the **:candidate** or **:writable-running** capabilities, then there will be no writable target, and an error will occur if these commands are entered.

All the high-level editing operations support the **\$\$default-operation** system variable. The <default-operation> parameter will be added the the NETCONF <edit-config> PDU if this variable is set to a value other than 'not-used'. The default is the enumeration 'none', which means do not use any default operation, and only use the explicit **nc:operation** attribute.

All the high-level editing operations support the **\$\$test-option** system variable. The <test-option> parameter will be added the the NETCONF <edit-config> PDU if this variable is set to a value other than 'none' (the default). This system variable will be used as the default if not entered directly.

```
replace /interfaces/interface[name='eth0']/ifMtu \
--test-option=$$test-option \
--value=$newvalue
```

This parameter can also be specified directly, each time the command is used.

```
$newvalue = 1518
```

```
replace /interfaces/interface[name='eth0']/ifMtu \
--test-option=test-only \
--value=$newvalue
```

## Yuma Tools User Manual

All the high-level retrieval operations support the **\$\$error-option** system variable. The <error-option> parameter will be added to the NETCONF <edit-config> PDU if this variable is set to a value other than 'none' (the default). This system variable will be used as the default if not entered directly.

```
replace /interfaces/interface[name='eth0']/ifMtu \\  
--error-option=$$error-option \  
--value=1518
```

This parameter can also be specified directly, each time the command is used.

```
replace /interfaces/interface[name='eth0']/ifMtu \  
--error-option=rollback-on-error \  
--value=1518
```

The high level **save** command is mapped to other commands, depending on the capabilities reported by the server.

### save command

capabilities	real command(s)
:candidate	commit
:writable-running	<none>
:startup	copy-config --source=running \ --target=startup

### 7.3.5 USING NOTIFICATIONS

The **create-subscription** command is used to start receiving notifications.

The **netconfd** server will include a <sequence-id> element in any notification that is saved in the replay buffer. This unsigned integer can be used to help debug notification filters (i.e., if non-consecutive <sequence-id> values are received, then the notification was filtered, or dropped due to access control policy).

If any replay notifications are desired, then the **--startTime** parameter must be included. At the end of the stored notifications, the server will send the <replayComplete> event. This notification type is not saved, and will not be found in the server replay buffer, if replay is supported by the server. The **netconfd** server will not include a <sequence-id> element in this notification type.

If the notification subscription should stop at a certain time, then the **--stopTime** parameter must be included. At the end of the stored notifications, the server will send the <replayComplete> event, followed by the <notificationComplete> event. . This notification type is not saved, and will not be found in the server replay buffer, if replay is supported by the server. The **netconfd** server will not include a <sequence-id> element in this notification type.

Notifications are printed to the log, using the current **\$\$display-mode** system variable setting, when and if they are received.

Notifications are also logged. Use the **eventlog** command to access the notifications stored in the event log.

### 7.3.6 CONFIGURATION PARAMETERS THAT AFFECT SESSIONS

The **--server**, **--user**, and **--password** configuration parameters can be used to start a NETCONF session automatically at startup time. The connect command will only be attempted at startup time if the **--server** parameter is present.

If all 3 of these parameters are present at startup time, then no interactive prompting for additional optional parameters will be done. Instead the connect command will be invoked right away.

During normal operation, the **--optional** configuration parameter, or the **\$\$optional** system variable, can be used to control interactive prompting for optional parameters.

The **--server** parameter is saved in the **\$\$server** system variable, which can be overwritten at any time. If set, this will be used as the initial default value for the **--server** parameter in the **connect** command.

The **--fixorder** configuration parameter can be used to control XML PDU ordering. If set to 'true', then the PDU will be reordered (if needed), to use the canonical order, according to the YANG specification. If 'false', the order parameters are entered at the command line will be their NETCONF PDU order as well. The default is 'true'. To send the server incorrectly ordered data structures on purpose, set this parameter to 'false'.

The **--user** parameter is saved in the **\$\$user** system variable, which can be overwritten at any time. If set, this will be used as the initial default value for the **--user** parameter in the **connect** command.

The **--with-defaults** configuration parameter, or the **\$\$with-defaults** system variable, can be used to set the default value for the 'with-defaults' parameter extension for the NETCONF **get**, **get-config**, and **copy-config** protocol operations. The default is 'none'.

The **--error-option** configuration parameter, or the **\$\$error-option** system parameter, can be used to set the default value for the **--error-option** parameter for the NETCONF edit-config protocol operation. The default is 'none'.

The **--test-option** configuration parameter, or the **\$\$test-option** system parameter, can be used to set the default value for the **--test-option** parameter for the NETCONF edit-config protocol operation. The default is 'none'.

The **--bad-data** configuration parameter, or the **\$\$bad-data** system variable, can be used to control how **yangcli** handles parameter values that are known to be invalid, or usage of optional protocol operations that the current session does not support. The default value is 'check'. To use **yangcli** in a testing mode to send the server incorrect data on purpose, set this parameter to 'warn' or 'ignore'.

### 7.3.7 TROUBLE-SHOOTING NETCONF SESSION PROBLEMS

If the NETCONF session does not start for any reason, one or more error messages will be printed, and the prompt will indicate 'idle' mode. This section assumes that the server is **netconfd**, and these debugging steps may not apply to all NETCONF agents.

#### If the NETCONF session does not start:

- make sure the server is reachable
  - try to 'ping' the server and see if it responds
- make sure the SSH server is responding
  - try to login in to the server using normal SSH login on port 22
- make sure a firewall is not blocking TCP port 830

## Yuma Tools User Manual

- try to connect to the NETCONF server using the **--port=22** option
- make sure the netconf-subsystem is configured correctly in /etc/ssh/sshd\_config  
there should be the proper configuration commands for NETCONF to work. For example, the **netconfd** server configuration might look like this:

```
Port 22
Port 830
Subsystem    netconf      /usr/sbin/netconf-subsystem
```

- make sure the **netconfd** server is running. Use the unix 'ps' command, or check the **netconfd** log file, to make sure it is running.

```
ps -alx | grep netconf
(look for 1 'netconfd' and N 'netconf-subsystem' -- 1 for
each active session)
```

- make sure the user name is correct
  - This must be a valid user name on the system
- make sure the password is correct
  - This must be the valid password (in /etc/passwd or /etc/shadow) for the specified user name

### If the NETCONF session stops responding:

- make sure the server is still reachable
  - try to 'ping' the server and see if it responds
- make sure the SSH server is still responding
  - try to login in to the server using normal SSH login on port 22

### If the NETCONF server is not accepting a certain command:

- make sure the command (or parameters used in the command) is actually supported by the server.
  - There may be features, when statements, or deviation statements that indicate the server does not support the command or one or more of its parameters.
- make sure that access control configured on the server is not blocking the command. The error-tag should be 'access-denied' in this case.

### If the NETCONF server is not returning the expected data in a <get> or <get-config> protocol operation::

- Make sure all the parameters are supported by the server
  - the **:xpath** capability must be advertised by the server to use the 'select' attribute in the <get> or <get-config> operations
  - the **:with-defaults** capability must be advertised by the server to use the <with-defaults> parameter

- if using a filter, try to retrieve the data without a filter and see if it is there
- make sure that access control configured on the server is not blocking the retrieval. There will not be any error reported in this case. The server will simply skip over any unauthorized data, and leave it out of the <rpc-reply>.
- set the logging level to debug2 or higher, and look closely at the PDUs being sent to the server. Set the display mode to a value other than 'plain' to make sure the correct namespaces are being used in the request.

### If an <edit-config> operation is failing unexpectedly:

- make sure that access control configured on the server is not blocking the request. The error-tag should be 'access-denied' in this case.
- make sure an unsupported parameter or parameter value is not used
  - <test-option> is not supported unless the **:validate** capability is advertised by the server
  - <error-option> = 'rollback-on-error' is not supported unless the **:rollback-on-error** capability is advertised by the server
- if the request contains an edit to a nested data structure, make sure the parent data structure(s) are in place as expected. The <default-operation> parameter is set to 'none' in the high level editing operations, so any data 'above' the edited data must already exist.
- set the logging level to debug2 or higher, and look closely at the PDUs being sent to the server. Set the display mode to a value other than 'plain' to make sure the correct namespaces are being used in the request.

## 7.4 Command Reference

This section describes all the **yangcli** local and remote commands built-in when using the **netconfd** server.

There may be more or less commands available, depending on the YANG modules actually loaded at the time.

The specific NETCONF capabilities needed are listed for each remote command. No capabilities are ever needed for a local command.

It is possible that other agents will support protocol operations that **netconfd** does not support. If yangcli has the YANG file available for the module, then it can be managed with the high level commands. Low-level commands can still be used with external data (e.g., @mydatafile.xml).

Any YANG rpc statement can be used as a remote **yangcli** command. Refer to the server vendor documentation for details on the protocol operations, database contents, and notification definitions that they support.

### 7.4.1 cd

The **cd** command is used to change the current working directory.

#### cd command

Command type:	local
---------------	-------

## Yuma Tools User Manual

Default parameter:	dir
Min parameters:	1
Max parameters:	1
Return type:	status
YANG file:	yangcli.yang

Command Parameters:

- **dir**
  - type: string
  - usage: mandatory
  - default: none
  - The '**dir**' string must contain a valid directory specification

Positive Response:

- the new current working directory is printed

Negative Response:

- an error message will be printed describing the error

Usage:

```
yangcli> cd ~/modules
```

```
Current working directory is /home/andy/modules
```

```
yangcli> cd --dir=$YUMA_HOME
```

```
Current working directory is /home/andy/swdev/yuma/trunk/netconf
```

```
yangcli>
```

### 7.4.2 CLOSE-SESSION

The **close-session** command is used to terminate the current NETCONF session. A NETCONF server should always accept this command if it is valid, and not reject it due to access control enforcement or if the server is in notification delivery mode.

#### **close-session command**

Command type:	remote
Default parameter:	none
Min parameters:	0
Max parameters:	0

## Yuma Tools User Manual

Return type:	status
YANG file:	yuma-netconf.yang

Command Parameters:

- none

Positive Response:

- the session is terminated and the command prompt is changed to indicate idle mode

Negative Response:

- an <rpc-error> message will be printed describing the error

Usage:

```
yangcli andy@myagent> close-session
```

```
RPC OK Reply 2 for session 10:
```

```
yangcli>
```

Reference:

- RFC 4741, section 7.8

### **7.4.3 COMMIT**

The **commit** command is used to save the edits in the <candidate> database into the <running> database. If there are no edits it will have no effect.

#### **commit command**

Command type:	remote
Default parameter:	none
Min parameters:	0
Max parameters:	2
Return type:	status
YANG file:	yuma-netconf.yang
Capabilities needed:	:candidate
Capabilities optional:	:confirmed-commit

Command Parameters:

- **confirmed**
  - type: empty
  - usage: optional
  - default: none
  - capabilities needed: :confirmed-commit

## Yuma Tools User Manual

- This parameter requests a confirmed commit procedure. The server will expect another **commit** command before the **confirm-timeout** time period expires.
- **confirm-timeout**
  - type: uint32 (range: 1 .. max)
  - usage: optional
  - default: 600 seconds
  - capabilities needed: :confirmed-commit
  - This is the number of seconds to request before the timeout.  
The '**confirmed**' leaf must also be present for this parameter to have any affect.

Positive Response:

- the session is terminated and the command prompt is changed to indicate idle mode

Negative Response:

- an <rpc-error> message will be printed describing the error

Usage:

```
yangcli andy@myagent> commit
```

```
RPC OK Reply 5 for session 10:
```

```
yangcli andy@myagent>
```

Reference:

- RFC 4741, section 8.3.4

### 7.4.4 CONNECT

The **connect** command is used to start a session with a NETCONF server.

If there already is a NETCONF session active, then an error message will be printed and the command will not be executed.

#### connect command

Command type:	remote
Default parameter:	server
Min parameters:	3
Max parameters:	5
Return type:	status
YANG file:	yangcli.yang

Command Parameters:

- **server**
  - type: inet:ip-address (string containing IP address or DNS name)

## Yuma Tools User Manual

- usage: mandatory
- default: previous server used, if any, will be presented as the default, but not used automatically
- This parameter specifies the server address for the session.
- **password**
  - type: string (ncx:password)
  - usage: mandatory
  - default: previous password used, if any, will be presented as the default, but not used automatically
  - This parameter specifies the password string to use to establish the session. It will not be echoed in parameter mode or saved in the command history buffer.
- **port**
  - type: uint16
  - usage: optional
  - default: 830
  - This parameter specifies the TCP port number that should be used for the session.
- **timeout**
  - type: uint32 (0 = no timeout, otherwise the number of seconds to wait)
  - usage: optional
  - default: set to the **\$\$timeout** system variable, default 30 seconds
  - This parameter specifies the number of seconds to wait for a response from the server before giving up. The session will not be dropped if a remote command times out, but any late response will be dropped. A value of zero means no timeout should be used, and **yangcli** will wait forever for a response.
- **user**
  - type: string
  - usage: mandatory
  - default: previous user name used, if any, will be presented as the default, but not used automatically
  - This parameter specifies the user name to use for the session. This must be a valid user name on the NETCONF server.

Positive Response:

- the session is started and the prompt changes to include the 'user@server' string.

Negative Response:

- One or more error messages will be printed. Refer to the section on trouble-shooting NETCONF Session problems for more details.

Usage:

```
yangcli> connect myagent user=andy password=yangrocks

<startup screen printed>
```

```
yangcli andy@myagent>
```

## 7.4.5 COPY-CONFIG

The **copy-config** command is used to copy one entire NETCONF database to another location.

Not all possible parameter combinations will be supported by every server. In fact, the NETCONF protocol does not require any parameters to be supported unless the **:startup** or **:url** capabilities is supported by the server.

If the server supports the **:startup** capability, then it must support:

```
yangcli andy@myagent> copy-config source=running target=startup
```

This is the standard way to save a snapshot of the current running configuration in non-volatile storage, if the server has a separate startup database. If not, the server will automatically save any changes to the running configuration to non-volatile storage.

### copy-config command

Command type:	remote
Default parameter:	none
Min parameters:	2
Max parameters:	3
Return type:	status
YANG file:	yuma-netconf.yang
Capabilities needed:	none
Capabilities optional:	:candidate :startup :url :with-defaults

Command 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
- **url**
  - type: yang:uri
  - capabilities needed: :url, and the scheme used in the URL must be specified in the :url capability 'schemes' parameter
  - To enter this parameter, the interactive mode must be used. The shorthand mode (source=url) cannot be used, since this parameter contains a string.
- **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
    - **running**
      - type: empty
      - capabilities needed: :writable-running (still optional to implement)
        - **netconfd** does not support this mode
    - **startup**
      - type: empty
      - capabilities needed: startup
    - **url**
      - type: yang:uri
      - capabilities needed: :url, and the scheme used in the URL must be specified in the :url capability 'schemes' parameter.
      - To enter this parameter, the interactive mode must be used. The shorthand mode (target=url) cannot be used, since this parameter contains a string.
- **with-defaults**
  - type: enumeration (none report-all trim explicit)
  - usage: optional

## Yuma Tools User Manual

- default: none
- capabilities needed: with-defaults
- This parameter controls how nodes containing only default values are copied to the target database.

Positive Response:

- the server returns <ok/>

Negative Response:

- An error message will be printed if errors are detected locally.
- An <rpc-error> message will be printed if the server detected an error.

Usage:

```
yangcli andy@myagent> copy-config source=candidate
```

Enter a number of the selected case statement:

```
1: case candidate:  
    leaf candidate  
2: case running:  
    leaf running  
3: case startup:  
    leaf startup  
4: case url:  
    leaf url
```

Enter choice number [1 - 4]:

```
yangcli andy@myagent:copy-config> 4
```

```
Filling optional case /copy-config/input/target/config-source/url
```

Enter string value for leaf <url>:

```
yangcli andy@myagent:copy-config> file:///configs/myconfig.xml
```

RPC OK Reply 12 for session 10:

```
yangcli andy@myagent>
```

Reference:

- RFC 4741, section 7.3

### 7.4.6 CREATE

The **create** command is a high-level <edit-config> operation. It is used to create some new nodes in the default target database.

A target node is specified (in 1 of 2 ways), and then the data structure is filled in. Only mandatory nodes will be filled in unless the **\$\$optional** system variable is set to 'true'.

## Yuma Tools User Manual

Refer to the **fill** command for more details on interactive mode data structure completion.

### create command

Command type:	remote
Default parameter:	target
Min parameters:	1
Max parameters:	5
Return type:	status
YANG file:	yangcli.yang
Capabilities needed:	:candidate or :writable-running

Command Parameters:

- **choice 'from'** (not entered)
  - type: choice of case 'varref' or case 'from-cli'
  - usage: mandatory
  - default: none
  - This parameter specifies the where **yangcli** should get the data from, for the create operation. It is either a user variable or from interactive input at the command line.
- **varref**
  - type: string
  - usage: mandatory
  - default: none
  - This parameter specifies the name of the user variable to use for the target of the create operation. The parameter must exist (e.g., created with the **fill** command) or an error message will be printed.
- **case from-cli** (not entered)
  - **target**
    - type: string
    - usage: mandatory
    - default: none
    - This parameter specifies the database target node of the create operation. This is an **ncx:schema-instance** string, so any instance identifier, or absolute path expression, or something in between, is accepted.
  - **optional**
    - type: empty
    - usage: optional
    - default: controlled by **\$\$optional** system variable

## Yuma Tools User Manual

- This parameter controls whether optional nodes within the target will be filled in. It can be used to override the **\$\$optional** system variable, when it is set to 'false'.
- **value**
  - type: anyxml
  - usage: mandatory
  - default: none
  - This parameter specifies the value that should be used for the contents of the **target** parameter. If it is entered, then the interactive mode prompting for missing parameters will be skipped, if this parameter is complete (or all mandatory nodes present if the **\$\$optional** system variable is 'false'). For example, if the target is a leaf, then specifying this parameter will always cause the interactive prompt mode to be skipped.
- **timeout**
  - type: uint32 (0 = no timeout, otherwise the number of seconds to wait)
  - usage: optional
  - default: set to the **\$\$timeout** system variable, default 30 seconds
  - This parameter specifies the number of seconds to wait for a response from the server before giving up. The session will not be dropped if a remote command times out, but any late response will be dropped. A value of zero means no timeout should be used, and yangcli will wait forever for a response.

System Variables:

- **\$\$default-operation**
  - The <default-operation> parameter for the <edit-config> operation will be derived from this variable.
- **\$\$error-option**
  - The <error-option> parameter for the <edit-config> operation will be derived from this variable
- **\$\$test-option**
  - The <test-option> parameter for the <edit-config> operation will be derived from this variable

Positive Response:

- the server returns <ok/>

Negative Response:

- An error message will be printed if errors are detected locally.
- An <rpc-error> message will be printed if the server detected an error.

Usage:

```
yangcli andy@myagent> create varref=myvar
```

RPC OK Reply 10 for session 10:

```
yangcli andy@myagent> create /nacm/rules/dataRule \
(user will be prompted to fill in the dataRule contents)
```

RPC OK Reply 11 for session 10:

```
yangcli andy@myagent> create \
    target=/nacm/rules/dataRule[name='test rule']/comment \
    value="this test rule is temporary. Do not remove!"
(no user prompting; <edit-config> request sent right away)
```

RPC OK Reply 12 for session 10:

```
yangcli andy@myagent>
```

Reference:

- RFC 4741, section 7.2

## 7.4.7 CREATE-SUBSCRIPTION

The create-subscription command is used to start receiving notifications from the server.

The :notification capability must be supported by the server to use this command.

Unless the :interleave capability is also supported by the server, then only the **close-session** command can be used while notifications are being delivered.

### create-subscription command

Command type:	remote
Default parameter:	none
Min parameters:	0
Max parameters:	4
Return type:	status
YANG file:	notifications.yang
Capabilities needed:	:notification
Capabilities optional:	:interleave

Command Parameters:

- **stream**
  - type: string
  - usage: optional
  - default: 'NETCONF'
  - This parameter specifies the name of the notification stream for this subscription request. Only the 'NETCONF' stream is mandatory to implement. Any other stream contains vendor-specific content, and may not be fully supported, depending on the stream encoding.
- **filter**

## Yuma Tools User Manual

- type: anyxml (same as the <get> or <get-config> filter parameter)
- usage: optional
- 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
  - usage: optional
  - default: none
  - 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.
  - After all the replay notifications are delivered, the server will send a <replayComplete> eventType, indicating there are no more replay notifications that match the subscription request.
- **stopTime**
  - type: yang:date-and-time
  - usage: optional (not allowed unless startTime is also present)
  - default: none
  - 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.
    - If this parameter is used, then the entire subscription will stop after this specified time, even if it is in the future. The <notificationComplete> eventType will be sent by the server when this event occurs.
    - If this parameter is not used (but **startTime** is used), then the server will continue to deliver 'live' notifications after the <replayComplete> eventType is sent by the server.

Positive Response:

- the server returns <ok/>

Negative Response:

- An error message will be printed if errors are detected locally.
- An <rpc-error> message will be printed if the server detected an error.

## Usage:

```
yangcli andy@myagent> create-subscription
```

RPC OK Reply 13 for session 10:

```
yangcli andy@myagent> create-subscription \  
startTime=2009-01-01T00:00:00Z
```

RPC OK Reply 14 for session 10:

```
yangcli andy@myagent>
```

## Reference:

- RFC 5277, section 2.1.1

## 7.4.8 **DELETE**

The **delete** command is a high-level <edit-config> operation. It is used to delete an existing subtree in the default target database.

A target node is specified, and then any missing key leafs (if any) within the data structure are filled in. If the target is a leaf-list, then the user will be prompted for the value of the leaf-list node to be deleted.

Refer to the **fill** command for more details on interactive mode data structure completion.

### **delete command**

Command type:	remote
Default parameter:	target
Min parameters:	1
Max parameters:	1
Return type:	status
YANG file:	yangcli.yang
Capabilities needed:	:candidate or :writable-running

### Command Parameters:

- **target**
  - type: string
  - usage: mandatory
  - default: none
  - This parameter specifies the database target node of the delete operation. This is an **ncx:schema-instance** string, so any instance identifier, or absolute path expression, or something in between, is accepted.

### System Variables:

- **\$\$default-operation**
  - The <default-operation> parameter for the <edit-config> operation will be derived from this variable.
- **\$\$error-option**
  - The <error-option> parameter for the <edit-config> operation will be derived from this variable
- **\$\$optional**
  - Controls whether optional descendant nodes will be filled into the **target** parameter contents
- **\$\$test-option**
  - The <test-option> parameter for the <edit-config> operation will be derived from this variable

### Positive Response:

- the server returns <ok/>

### Negative Response:

- An error message will be printed if errors are detected locally.
- An <rpc-error> message will be printed if the server detected an error.

### Usage:

```
yangcli andy@myagent> delete /nacm/rules/dataRule \
(user will be prompted to fill in the dataRule 'name' key leaf)
```

RPC OK Reply 15 for session 10:

```
yangcli andy@myagent> delete \
    target=/nacm/rules/dataRule[name='test rule']/comment
(no user prompting; <edit-config> request sent right away)
```

RPC OK Reply 16 for session 10:

```
yangcli
```

### Reference:

- RFC 4741, section 7.2

## 7.4.9 DELETE-CONFIG

The **delete-config** command is used to delete an entire NETCONF database.

Not all possible **target** parameter values will be supported by every server. In fact, the NETCONF protocol does not require that any database be supported by this operation.

If the server supports the :url capability, then it may support deletion of local file databases in this manner.:

### **delete-config command**

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

Command Parameters:

- **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 delete operation.
  - container contents: 1 of N:
    - **startup**
      - type: empty
      - capabilities needed: startup
      - a server may support this target
    - **url**
      - type: yang:uri
      - capabilities needed: :url, and the scheme used in the URL must be specified in the :url capability 'schemes' parameter.
      - To enter this parameter, the interactive mode must be used. The shorthand mode (target=url) cannot be used, since this parameter contains a string.
      - a server may support this parameter

Positive Response:

- the server returns <ok/>

Negative Response:

- An error message will be printed if errors are detected locally.
- An <rpc-error> message will be printed if the server detected an error.

Usage:

## Yuma Tools User Manual

```
yangcli andy@myagent> delete-config target= startup  
  
RPC OK Reply 17 for session 10:  
  
yangcli andy@myagent>
```

Reference:

- RFC 4741, section 7.4

### 7.4.10 DISCARD-CHANGES

The **discard-changes** command is used to delete any edits that exist in the <candidate> database, on the NETCONF server. The server will only accept this command if the :candidate capability is supported. If the <candidate> database is locked by another session, then this request will fail with an 'in-use' error.

#### **discard-changes command**

Command type:	remote
Default parameter:	none
Min parameters:	0
Max parameters:	0
Return type:	status
YANG file:	yuma-netconf.yang
Capabilities needed:	:candidate

Command Parameters: none

Positive Response:

- the server returns <ok/>

Negative Response:

- An error message will be printed if errors are detected locally.
- An <rpc-error> message will be printed if the server detected an error.

Usage:

```
yangcli andy@myagent> discard-changes  
  
RPC OK Reply 18 for session 10:  
  
yangcli andy@myagent>
```

Reference:

- RFC 4741, section 8.3.4.2

## 7.4.11 EDIT-CONFIG

The edit-config command allows a subset of a NETCONF database on the server to be changed. If the server supports the :url capability, then it may support editing of local file databases. If the server supports the :candidate capability, then it will allow edits to the <candidate> database. If the server supports the :writable-running capability, it will support edits to the <running> database. It is not likely that a server will support the <candidate> and <running> database as targets at the same time, since changes to the <running> configuration would not be reflected in the <candidate> database, while it was being edited by a different session.

### edit-config command

Command type:	remote
Default parameter:	none
Min parameters:	2
Max parameters:	5
Return type:	status
YANG file:	yuma-netconf.yang
Capabilities needed:	:candidate or :writable-running
Capabilities optional:	:url :rollback-on-error :validate

Command Parameters:

- **default-operation**
  - type: enumeration (merge replace none)
  - usage: optional
  - 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.
    - The high-level edit operations provided by **yangcli** will set this parameter to 'none'. This is the safest value, since only subtrees that have an explicit **nc:operation** attribute in effect can possibly be altered by the command.
    - If the value is 'merge', then any missing nodes in the database will be automatically created as needed.
    - If the value is 'replace', then the target database will be pruned to match the edits, as needed. Only the data from the **config** parameter will remain if this value is used. (Use with extreme caution).
- **error-option**
  - type: enumeration (stop-on-error continue-on-error rollback-on-error)

## Yuma Tools User Manual

- usage: optional
- default: stop-on-error
- This parameter specifies what the server should do when an error is encountered.
  - The rollback-on-error value is only allowed if the **:rollback-on-error** capability is supported by the server.
  - The standard is not clear what continue-on-error really means. It is suggested that this value not be used. It is possible that the server will validate all input parameters before making any changes, no matter how this parameter is set.
- **choice edit-content** (not entered)
  - **config**
    - type: anyxml
    - usage: mandatory
    - default: none
    - This parameter specifies the subset of the database that should be changed. This is the most common way to edit a NETCONF server database, since it is mandatory to support by all agents.
  - **url**
    - type: yang:uri
    - capabilities needed: **:url**, and the scheme used in the URL must be specified in the **:url** capability 'schemes' parameter.
    - To enter this parameter, the interactive mode must be used. The shorthand mode (target=url) cannot be used, since this parameter contains a string.
- **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 edit operation.
  - container contents: choice: 1 of N:
    - **candidate**
      - type: empty
      - capabilities needed: **:candidate**
    - **running**
      - type: empty
      - capabilities needed: **:writable-running**
- **test-option**
  - type: enumeration (test-then-set set test-only)
  - usage: optional
  - default: set (unless **:validate** capability is supported, the 'test-then-set' is the default value)
  - This parameter specifies how the server should test the **edit-content** parameter before using it.

## Yuma Tools User Manual

- If the value is 'set' (normal case), the server will apply validation tests as needed for the individual data structures being edited
- The value 'test-then-set' is only allowed if the :validate capability is supported by the server. The server will test if the entire database will be valid after the edits are made, before making any changes to the candidate configuration.
  - This mode is very resource intensive. Set this parameter to 'set' for better performance, and run the validation tests manually with the **validate** command.
- The value 'test-only' is not supported by all agents. It will be in the next version of the NETCONF protocol, but is non-standard at this time.
  - Use this value to check if a specific edit should succeed or not, allowing errors to be corrected before altering the database for real.

Positive Response:

- the server returns <ok/>

Negative Response:

- An error message will be printed if errors are detected locally.
- An <rpc-error> message will be printed if the server detected an error.

Usage:

```
yangcli andy@myagent> edit-config target=candidate \
    default-operation=merge \
    test-option=test \
    error-option=stop-on-error \
    config=@myconfig.xml
```

RPC OK Reply 19 for session 10:

```
yangcli andy@myagent>
```

Reference:

- RFC 4741, section 7.2

### 7.4.12 EVENTLOG

The **eventlog** command is used to view or clear all or part of the notification event log. This log will be empty if no well-formed notifications have been received from any server.

The **eventlog show** command is used to display some event log entries.

The **eventlog clear** command is used to delete some event log entries.

If no parameters are entered, it is the same as entering 'eventlog show=-1'.

The event log is not automatically emptied when a session terminates, in case the session was dropped unexpectedly. New entries will be appended to the event log as new sessions and/or subscriptions are started.

**eventlog command**

Command type:	local
Default parameter:	show
Min parameters:	0
Max parameters:	3
Return type:	status
YANG file:	yangcli.yang

Command Parameters:

- **choice eventlog-action** (not entered):
  - type: choice of case 'show-case' or leaf 'clear'
  - usage: optional
  - default: show=-1 is used as the default if nothing entered
  - This parameter specifies the event log action that should be performed.
    - **clear**
      - type: int32 (-1 to clear all entries; 1 to max to delete N entries)
      - usage: optional
      - default: -1
      - This parameter specifies the maximum number of event log entries to be deleted, starting from the oldest entries in the event log. The value -1 means delete all the entries. Otherwise the value must be greater than zero, and up to that many entries will be deleted.
    - **case show-case** (not entered)
      - **choice help-mode** (not entered) (default choice is 'normal')
        - **brief**
          - type: empty
          - usage: optional
          - default: none
          - This parameter specifies that brief documentation mode should be used. The event log index, sequence ID, and <eventType> will be displayed in this mode.
        - **normal**
          - type: empty
          - usage: optional
          - default: none
          - This parameter specifies that normal documentation mode should be used. The event log index, <eventTime>, sequence ID, and <eventType> will be displayed in this mode.
        - **full**
          - type: empty

## Yuma Tools User Manual

- usage: optional
- default: none
- This parameter specifies that full documentation mode should be used. The event log index, <eventTime>, sequence ID, and <eventType> will be displayed in this mode. In addition, the entire contents of the notification PDU will be displayed, using the current **\$\$display-mode** setting.
- **show**
  - type: int32 (-1 for all, 1 to max for N entries)
  - usage: optional
  - default: -1
  - This parameter specifies the number of event log entries that should be displayed. The value '-1' indicates all the entries should be displayed. Otherwise, the value must be greater than zero, indicating the number of entries to display.
- **start**
  - type: uint32
  - usage: optional
  - default: 0
  - This parameter specifies the start position in the event log to start displaying entries. The first entry is number zero. Each time the event log is cleared, the numbering restarts.

System Variables:

- **\$\$display-mode**
  - The log entries printed when help-mode='full' are formatted according to the current value of this system variable.

Positive Response:

- the event log entries are printed or cleared as requested

Negative Response:

- An error message will be printed if errors are detected.

Usage:

```
yangcli andy@myagent> eventlog show=5 start=3

[3] [2009-07-10T02:21:10Z] (4)      <sysSessionStart>
[4] [2009-07-10T02:23:14Z] (5)      <sysSessionEnd>
[5] [2009-07-10T02:23:23Z] (6)      <sysSessionStart>
[6] [2009-07-10T02:24:52Z] (7)      <sysConfigChange>
[7] [2009-07-10T02:24:57Z] (8)      <sysSessionEnd>

yangcli andy@myagent>
```

### 7.4.13 FILL

The **fill** command is used to create a user variable for reuse in other commands.

## Yuma Tools User Manual

It is used in an assignment statement to create a variable from various sources.

If it is not used in an assignment statement, then the result will not be saved, so the command will have no effect in this case.

The value contents will mirror the subtree within the NETCONF database indicated by the target parameter. If not completely provided, then missing descendant nodes will be filled in interactively, by prompting for each missing node.

### **fill command**

Command type:	local
Default parameter:	target
Min parameters:	2
Max parameters:	3
Return type:	data
YANG file:	yangcli.yang

Command Parameters:

- **optional**
  - type: empty
  - usage: optional
  - default: controlled by **\$\$optional** system variable
  - This parameter controls whether optional nodes within the target will be filled in. It can be used to override the **\$\$optional** system variable, when it is set to 'false'.
- **target**
  - type: string
  - usage: mandatory
  - default: none
  - This parameter specifies the database target node of the create operation. This is an **ncx:schema-instance** string, so any instance identifier, or absolute path expression, or something in between, is accepted.
- **value**
  - type: anyxml
  - usage: mandatory
  - default: none
  - This parameter specifies the content to use for the filled variable.
    - If this parameter is not entered, then the user will be prompted interactively to fill in the **target** data structure.
    - If a string is entered, then the target value being filled must be a leaf or leaf-list.
    - If a variable reference is entered, then it will be used as the content, if the target value being filled is a leaf or a leaf-list.

## Yuma Tools User Manual

- If the target value is a complex object, then the referenced variable must also be a complex object of the same type.
- An error will be reported if the global or local variable does not reference the same object type as the target parameter.

System Variables:

- `$$optional`
  - Controls whether optional descendant nodes will be filled into the **target** parameter contents

Positive Response:

- OK

Negative Response:

- An error message will be printed if errors are detected.

Output:

- data
  - type: anyxml
  - The data structure will mirror the requested target object.
  - The variable (if any) will retain the target object name and namespace so it can be used in other operations more easily. In the example below, the **\$my\_interface** local variable will have the module name 'interfaces' and name 'interface', when used in other commands such as **create** or **merge**.

Usage:

```
yangcli> $my-interface = fill \
           target=/interfaces/interface optional
           (user will be prompted to fill in all fields
            of the <interface> element)
OK

yangcli>
```

### 7.4.14 GET

The **get** command is used to retrieve data from the server.

#### **get command**

Command type:	remote
Default parameter:	none
Min parameters:	0
Max parameters:	2
Return type:	data

## Yuma Tools User Manual

YANG file:	yuma-netconf.yang
------------	-------------------

Command Parameters:

- **filter**
  - type: anyxml
  - usage: optional
  - default: none
  - This parameter specifies a boolean filter that should be applied to the stream. Any data in the <running> database (or non-config data) that does not match the filter will be left out of the <rpc-reply> response.
    - If no filter is used, the server will return the entire <running> database and all non-config data as well. This could be a lot of data, depending on the server.
    - If the result is empty after the filter is applied to the available data, then the server will send an empty <data> element in the <rpc-reply>
    - It is possible that access control will cause the <rpc-reply> to be pruned. The **netconfd** server will silently prune any unauthorized payload from the <rpc-reply>.
- **with-defaults**
  - type: enumeration (none report-all trim explicit)
  - usage: optional
  - default: none
  - capabilities needed: with-defaults
  - This parameter controls how nodes containing only default values are returned in the <rpc-reply>.

Positive Response:

- the server returns <data>

Negative Response:

- An error message will be printed if errors are detected locally.
- An <rpc-error> message will be printed if the server detected an error.

Output:

- **data**
  - type: anyxml
  - This element will contain the requested data from the <running> database, or non-config data from the server instrumentation.

Usage:

```
yangcli andy@myagent> get
```

```
RPC Data Reply 20 for session 10:
```

```
rpc-reply {
```

## Yuma Tools User Manual

```
data {  
    ... data returned by the server  
}  
}  
  
yangcli andy@myagent> get filter=@myfilter.xml  
  
RPC Data Reply 21 for session 10:  
  
rpc-reply {  
    data {  
    }  
}  
  
(the previous response will occur if the filter did not  
match anything or the server access control filtered the  
entire response)  
  
yangcli andy@myagent>
```

Reference:

- RFC 4741, section 7.7

### 7.4.15 GET-CONFIG

The **get-config** command is used to retrieve configuration data from the server.

#### get-config command

Command type:	remote
Default parameter:	none
Min parameters:	1
Max parameters:	3
Return type:	data
YANG file:	yuma-netconf.yang

Command Parameters:

- **filter**
  - type: anyxml
  - usage: optional
  - default: none

## Yuma Tools User Manual

- This parameter specifies a boolean filter that should be applied to the stream. Any data in the <running> database (or non-config data) that does not match the filter will be left out of the <rpc-reply> response.
  - If no filter is used, the server will return the entire <running> database and all non-config data as well. This could be a lot of data, depending on the server.
  - If the result is empty after the filter is applied to the available data, then the server will send an empty <data> element in the <rpc-reply>
  - It is possible that access control will cause the <rpc-reply> to be pruned. The **netconfd** server will silently prune any unauthorized payload from the <rpc-reply>.
- **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 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
    - **url**
      - type: yang:uri
      - capabilities needed: :url, and the scheme used in the URL must be specified in the :url capability 'schemes' parameter
      - To enter this parameter, the interactive mode must be used. The shorthand mode (source=url) cannot be used, since this parameter contains a string.
- **with-defaults**
  - type: enumeration (none report-all trim explicit)
  - usage: optional
  - default: none
  - capabilities needed: with-defaults
  - This parameter controls how nodes containing only default values are returned in the <rpc-reply>.

Positive Response:

- the server returns <data>

Negative Response:

## Yuma Tools User Manual

- An error message will be printed if errors are detected locally.
- An <rpc-error> message will be printed if the server detected an error.

Output:

- **data**
  - type: anyxml
  - This element will contain the requested data from the **source** database.

Usage:

```
yangcli andy@myagent> $my-config = get-config target=running
```

```
RPC Data Reply 22 for session 10:
```

```
rpc-reply {  
    data {  
        ... entire database returned by the server  
    }  
}
```

```
yangcli andy@myagent> @saved-config.xml = get-config \  
    filter=@myfilter.xml \  
    target=candidate
```

```
rpc-reply {  
    data {  
        ... data requested by the filter  
    }  
}
```

```
yangcli andy@myagent>
```

Reference:

- RFC 4741, section 7.1

### 7.4.16 GET-LOCKS

The **get-locks** command is a high-level wrapper for the <lock> operation. It is used to lock all the databases (<running> plus <candidate> and/or <startup> if they exist). If all the locks cannot be obtained, then release all the locks that were obtained (all-or-nothing).

The entire time to wait for a lock in use is set with the lock-timeout parameter.

The retry-interval parameter is used when the <lock> operation fails with a 'lock-denied' error-tag, because some other session has the lock.

## Yuma Tools User Manual

If the <candidate> cannot be locked for another reason, a <discard-changes> operation will be attempted to clear any leftover edits.

Normally, the errors received while attempting to acquire locks are not printed to the log, like normal commands. Instead, if \$\$log-level system parameter is set to 'debug2' or 'debug3', then these messages will be printed.

### get-locks command

Command type:	remote
Default parameter:	none
Min parameters:	0
Max parameters:	3
Return type:	status
YANG file:	yangcli.yang

Command Parameters:

- **lock-timeout**
  - type: uint32 (seconds)
  - usage: optional
  - default: 120 seconds (2 minutes)
  - This parameter specifies how long to wait for a lock that is in use by another session.
- **retry-interval**
  - type: uint32 (seconds)
  - usage: optional
  - default: 2 seconds
  - This parameter specifies how long to wait to retry for a lock.
- **cleanup**
  - type:boolean
  - usage: optional
  - default: true
  - This parameter controls whether the 'release-locks' command will be called automatically if the entire set of required locks cannot be granted.

Positive Response:

- the server returns <ok/>

Negative Response:

- An error message will be printed if errors are detected locally.
- An <rpc-error> message will be printed if the server detected an error.

Usage:

## Yuma Tools User Manual

```
yangcli andy@myagent> get-locks lock-timeout=0

Sending <lock> operations for get-locks...

RPC OK Reply 6 for session 23:

RPC OK Reply 7 for session 23:

get-locks finished OK
yangcli andy@myagent>
```

### 7.4.17 GET-MY-SESSION

The **get-my-session** command is used to retrieve the session customization parameters from the server. It is only supported by the **netconfd** server.

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

#### **get-my-session command**

Command type:	remote
Default parameter:	none
Min parameters:	0
Max parameters:	0
Return type:	data
YANG file:	mysession.yang
Capabilities needed:	none

Command Parameters:

- none

Positive Response:

- the server returns <indent>, <linesize>, and <with-defaults> elements

Negative Response:

- An error message will be printed if errors are detected locally.
- An <rpc-error> message will be printed if the server detected an error.

Output:

- **indent**
  - type: uint32 (range 0 to 9)
  -
- **linesize**

- type: uint32
- 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 filtering behavior for the session.

```

yangcli andy@myagent> get-my-session
RPC Data Reply 25 for session 10:
rpc-reply {
    data {
        indent 3
        linesize 72
        with-defaults report-all
    }
}

```

yangcli andy@myagent>

### 7.4.18 GET-SCHEMA

The **get-schema** command is used to retrieve data model definition files from the server. This is part of the NETCONF monitoring draft. The server must support the **:schema-retrieval** capability to use this command.

If the server reports a module or module version that **yangcli** cannot find in its local module library, then an error message will be printed. The **get-schema** command can then be used to retrieve the missing module from the server.

The **ietf-netconf-state.yang** module includes a list of the schema supported by the server, which can be retrieved from a server that supports this module, such as **netconfd**.

```
yangcli andy@myagent> sget /ietf-netconf-state/schemas
```

The preceding command will return a <schemas> container with several <schema> child nodes. One example entry is shown below:

```

schemas {
    schema system 2009-06-04 YANG {
        identifier system
        version 2009-06-04
        format YANG
        namespace http://netconfcentral.com/ns/system
        location NETCONF
    }
}

```

## Yuma Tools User Manual

The <identifier>, <version> and <format> leafs can be used as the corresponding parameter values for the **get-schema** command. See the example below for more details.

### get-schema command

Command type:	remote
Default parameter:	none
Min parameters:	3
Max parameters:	3
Return type:	data
YANG file:	ietf-netconf-state.yang
Capabilities needed:	:schema-retrieval

Command Parameters:

- **identifier**
  - type: string
  - usage: mandatory
  - default: none
  - This parameter specifies the name of the module to retrieve.
    - Do not use any path specification or file extension; just the module name is entered.
    - The name is case-sensitive, and must be specified exactly as defined.
- **version**
  - type: string
  - usage: mandatory
  - default: none
  - This parameter specifies the version of the module to retrieve.
    - For YANG modules, this will be the most recent 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)
  - usage: mandatory
  - default: none
  - This parameter specifies the format of the module to be retrieved.
    - XSD: W3C REC REC-xmlschema-1-20041028
    - YANG: draft-ietf-netmod-yang
    - RNG: ISO/IEC 19757-2

## Yuma Tools User Manual

- **netconfd** only supports the YANG format

Positive Response:

- the server returns <data>

Negative Response:

- An error message will be printed if errors are detected locally.
- An <rpc-error> message will be printed if the server detected an error.

Output:

- data
  - type: anyxml
  - **yangcli** will strip off this XML container if the command result is being saved to a text file. Only the YANG contents will be saved instead.

Usage:

```
yangcli andy@myagent> @notifications.yang = get-schema \
    identifier=notifications \
    version=2009-06-04 \
    format=YANG
```

RPC Data Reply 24 for session 10:

```
rpc-reply {
    data {
        ... entire notifications.yang contents
    }
}
```

(after retrieval, the module can be loaded locally  
with the mgrload command)

```
yangcli andy@myagent> mgrload notifications.yang
```

OK

```
yangcli andy@myagent>
```

Reference:

- draft-ietf-netconf-monitoring-06.txt

### 7.4.19 HELP

The help command is used to print documentation to STDOUT.

## Yuma Tools User Manual

If no session is active, then only help for the local commands and the standard NETCONF commands will be available.

If a NETCONF session is active, then the documentation shown will attempt to exactly match the capabilities of the server.

If additional (i.e., augment generated) parameters are available, then they will be shown in the command output. If the server does not implement some parameters (e.g., feature not supported) then these parameters will not be shown in the command output.

If the server has modified an object with deviation statements, then the altered object will be shown.

The **ncx:hidden** extension suppresses the **help** command. If this extension is present in the YANG definition associated with the request, then no help will be available for that object or command.

### **help command**

Command type:	local
Default parameter:	command
Min parameters:	3
Max parameters:	3
Return type:	data
YANG file:	ietf-netconf-state.yang
Capabilities needed:	:schema-retrieval

Command Parameters:

- **choice helptype** (not entered)
  - **command**
    - type: string
    - usage: mandatory
    - default: none
    - This parameter specifies the name of the command for which documentation is requested
  - **commands**
    - type: empty
    - usage: mandatory
    - default: none
    - This parameter will request documentation for all available commands
  - **notification**
    - type: string
    - usage: mandatory
    - default: none
    - This parameter specifies the name of the notification for which documentation is requested

- **object**
  - type: string
  - usage: mandatory
  - default: none
  - This parameter specifies the name of the NETCONF database object for which documentation is requested.
    - Only top level objects are supported by this command.
    - Documentation for the entire object subtree will be printed, if the object is a container, choice, or list.
    - Documentation for nested objects is only available in parameter mode, using the escape commands for help ('?') and full help ('??')
- **type**
  - type: string
  - usage: mandatory
  - default: none
  - This parameter specifies the name of the YANG typedef for which documentation is requested
  - Only top-level typedefs are supported by this command. Local typedefs within groupings, containers, or lists are not exportable in YANG.
- **choice help-mode** (not entered) (default choice is 'normal')
  - **brief**
    - type: empty
    - usage: optional
    - default: none
    - This parameter specifies that brief documentation mode should be used.
  - **normal**
    - type: empty
    - usage: optional
    - default: none
    - This parameter specifies that normal documentation mode should be used.
  - **full**
    - type: empty
    - usage: optional
    - default: none
    - This parameter specifies that full documentation mode should be used.

Positive Response:

- the server prints the requested help text

Negative Response:

- An error message will be printed if errors are detected.

## Yuma Tools User Manual

### Usage:

```
yangcli> help help full
```

```
help
```

```
Print the yangcli help text
```

```
input
```

```
    default parameter: command
```

```
choice helptype
```

```
leaf command [NcxIdentifier]
```

```
    Show help for the specified command,
```

```
    also called an RPC method
```

```
leaf commands [empty]
```

```
    Show info for all local commands
```

```
leaf notification [NcxIdentifier]
```

```
    Show help for the specified notification
```

```
leaf object [NcxIdentifier]
```

```
    Show help for the specified object
```

```
leaf type [NcxIdentifier]
```

```
    Show help for the specified type
```

```
choice help-mode
```

```
leaf brief [empty]
```

```
    Show brief help text
```

```
leaf normal [empty]
```

```
    Show normal help text
```

```
leaf full [empty]
```

```
    Show full help text
```

```
yangcli andy@myagent> help notification sysConfigChange
```

```
notification sysConfigChange
```

```
    Generated when the <running> configuration is changed.
```

```
leaf userName [string]
```

```
leaf sessionId [SessionId]
```

```
    range: 1..max
```

```

leaf remoteHost [ip-address]

leaf target [string]

leaf operation [EditOperationType] [d:merge]
    enum values: merge replace create delete

yangcli andy@svnserver>

```

## 7.4.20 HISTORY

The **history** command is used to show, clear, load, or save the command line history buffer.

Use the **recall** command to recall a previously executed command line, after getting the line number from the **history show** command.

All lines entered will be saved in the history buffer except an **ncx:password** value entered in parameter mode.

When **yangcli** starts, the command line history buffer is empty. If a history file was previously stored with the **history save** command, then it can be recalled into the buffer with the **history load** command.

The **history clear** command is used to delete the entire command line history buffer.

The numbering sequence for commands, starts from zero and keeps incremented until the program exits. If the history buffer is cleared, then the number sequence will continue, not start over at zero.

### **history command**

Command type:	local
Default parameter:	show
Min parameters:	0
Max parameters:	2
Return type:	data
YANG file:	yangcli.yang

Command Parameters:

- **choice history-action** (not entered)
  - **clear**
    - type: empty
    - usage: optional
    - default: none
    - This parameter specifies that the history buffer should be cleared. Unless the contents have been saved with the **history save** command, there is no way to recover the cleared buffer contents after this command is executed.

- **load**
  - type: string
  - usage: optional
  - default: \$HOME/.yangcli\_history
  - This parameter specifies a command history file, and causes the current command line history contents to be loaded from that file.
  - Special processing for this command allows the history file to be omitted in idle mode, even though the load parameter is not type 'empty'.
 

```
yangcli> history load
      same as:
yangcli> history load=$HOME/.yangcli_history
```
- **save**
  - type: string
  - usage: optional
  - default: \$HOME/.yangcli\_history
  - This parameter specifies a command history file, and causes the current command line history contents to be saved to that file.
  - Special processing for this command allows the history file to be omitted in idle mode, even though the save parameter is not type 'empty'.
 

```
yangcli> history save
      same as:
yangcli> history save=$HOME/.yangcli_history
```
- **show**
  - type: int32 (-1 for all entries; 1..max for N entries)
  - usage: optional
  - default: -1
  - This parameter specifies the maximum number of history entries to show.
    - If no case is selected from this choice, then the command '**history show=-1**' will be used by default.
    - The **help-mode** choice parameter is only used with the **history show** command.
      - If the **--brief** or **--normal** modes are selected the the format will include the command number and the command line.
      - If the **--full** mode is selected, then the command data and time will also be printed.
- **choice help-mode** (not entered)
 

This parameter is ignored unless the **history show** command is entered.

  - **brief**
    - type: empty
    - usage: optional
    - default: none
    - This parameter specifies that brief documentation mode should be used.

- **normal**
  - type: empty
  - usage: optional
  - default: none
  - This parameter specifies that normal documentation mode should be used.
- **full**
  - type: empty
  - usage: optional
  - default: none
  - This parameter specifies that full documentation mode should be used.

Positive Response:

- the requested history entries will be printed for the **history show** command
- all other commands will return OK

Negative Response:

- An error message will be printed if errors are detected.

Usage:

```
yangcli> history show=3 full
[27] 2009-07-04 09:25:34 sget /system --nofill
[28] 2009-07-04 09:34:17 @myconfig = get-config source=running
[29] 2009-07-04 09:43:54 history show=3 full

yangcli> history save=~/my-temp-history-file
OK
yangcli>
```

### 7.4.21 INSERT

The insert command is used to insert or move YANG list or leaf-list data into a NETCONF database. It is a high level command with utilizes the YANG 'insert' extensions to the NETCONF <edit-config> operation.

#### **insert command**

Command type:	remote
Default parameter:	target
Min parameters:	2
Max parameters:	7
Return type:	status
YANG file:	yangcli.yang

## Command Parameters:

- **choice 'from'** (not entered)
  - type: choice of case 'varref' or case 'from-cli'
  - usage: mandatory
  - default: none
  - This parameter specifies the where **yangcli** should get the data from, for the insert operation. It is either a user variable or from interactive input at the command line.
- **varref**
  - type: string
  - usage: mandatory
  - default: none
  - This parameter specifies the name of the user variable to use for the target of the insert operation. The parameter must exist (e.g., created with the **fill** command) or an error message will be printed.
- **case from-cli** (not entered)
  - **target**
    - type: string
    - usage: mandatory
    - default: none
    - This parameter specifies the database target node of the insert operation. This is an **ncx:schema-instance** string, so any instance identifier, or absolute path expression, or something in between, is accepted.
  - **optional**
    - type: empty
    - usage: optional
    - default: controlled by **\$\$optional** system variable
    - This parameter controls whether optional nodes within the target will be filled in. It can be used to override the **\$\$optional** system variable, when it is set to 'false'.
  - **value**
    - type: anyxml
    - usage: mandatory
    - default: none
    - This parameter specifies the value that should be used for the contents of the **target** parameter. If it is entered, then the interactive mode prompting for missing parameters will be skipped, if this parameter is complete (or all mandatory nodes present if the **\$\$optional** system variable is 'false'). For example, if the target is a leaf, then specifying this parameter will always cause the interactive prompt mode to be skipped.
- **edit-target**
  - type: string
  - usage: optional (must be present if the order parameter is set to 'before' or 'after').

## Yuma Tools User Manual

- default: none
- This parameter specifies the value or key clause that should be used, as the list or leaf-list insertion point. It identifies the existing entry that the new entry will be inserted before or after, depending on the **order** parameter.
  - For a leaf-list, the edit-target contains the value of the target leaf-list node within the configuration being edited. E.g., edit-target='fred'.
  - For a list, the edit-target contains the key values of the target list node within the configuration being edited. E.g., edit-target=[name='fred'][zipcode=90210].
- **order**
  - type: enumeration (first last before after)
  - usage: optional
  - default: last
  - The insert order that should be used. If the value 'before' or 'after' is selected, then the **edit-target** parameter must also be present.
- **operation**
  - type: enumeration (create merge replace)
  - usage: optional
  - default: merge
  - This parameter specifies the **nc:operation** attribute value for the NETCONF <edit-config> operation. The insert operation is secondary to the NETCONF operation attribute.
- **timeout**
  - type: uint32 (0 = no timeout, otherwise the number of seconds to wait)
  - usage: optional
  - default: set to the **\$\$timeout** system variable, default 30 seconds
  - This parameter specifies the number of seconds to wait for a response from the server before giving up. The session will not be dropped if a remote command times out, but any late response will be dropped. A value of zero means no timeout should be used, and yangcli will wait forever for a response.

System Variables:

- **\$\$default-operation**
  - The <default-operation> parameter for the <edit-config> operation will be derived from this variable.
- **\$\$error-option**
  - The <error-option> parameter for the <edit-config> operation will be derived from this variable
- **\$\$test-option**
  - The <test-option> parameter for the <edit-config> operation will be derived from this variable

Positive Response:

- the server returns <ok/>

Negative Response:

## Yuma Tools User Manual

- An error message will be printed if errors are detected locally.
- An <rpc-error> message will be printed if the server detected an error.

Usage:

```
yangcli andy@myagent> insert varref=myvar order=first
```

```
RPC OK Reply 25 for session 10:
```

```
yangcli andy@myagent> insert /nacm/rules/dataRule \
    order=after \
    edit-target="[name='test-rule']"
```

```
(user will be prompted to fill in the dataRule contents)
```

```
RPC OK Reply 26 for session 10:
```

```
yangcli andy@myagent>
```

Reference:

- draft-ietf-netmod-yang-09

### 7.4.22 KILL-SESSION

The **kill-session** command is used to terminate a NETCONF session (other than the current session). All NETCONF implementations must support this command. It is needed sometimes to unlock a NETCONF database locked by a session that is idle or stuck.

If the **lock** command returns a 'lock-denied' <error-tag>, it will also include an <error-info> field called <session-id>. This is the session number that currently holds the requested lock. The same value should be used for the **session-id** parameter in this command, to terminate the session will the lock.

Note: this is an extreme measure, which should be used with caution.

#### kill-session command

Command type:	remote
Default parameter:	target
Min parameters:	1
Max parameters:	1
Return type:	status
YANG file:	yuma-netconf.yang

Command Parameters:

- **session-id**
  - type: uint32 (range: 1.. max)

## Yuma Tools User Manual

- usage: mandatory
- default: none
- This parameter specifies the session number of the currently active NETCONF session that should be terminated.

Positive Response:

- the server returns <ok/>

Negative Response:

- An error message will be printed if errors are detected locally.
- An <rpc-error> message will be printed if the server detected an error.

Usage:

```
yangcli andy@myagent> kill-session session-id=11
```

```
RPC OK Reply 27 for session 10:
```

```
yangcli andy@myagent>
```

Reference:

- RFC 4741, section 7.9

### 7.4.23 LIST

This **list** command is used to display the commands, objects, and oids (object identifiers) that are available at the time.

The **list commands** command will display the local commands and the remote commands that are available in the current NETCONF session, or which have been loaded with the **mgrload** command.

The **list files** command will display the data files that are in the current data search path. The module parameter has no affect in this mode.

The **list modules** command will display the YANG files that are in the current YANG module search path. The module parameter has no affect in this mode.

The **list objects** command will display the top-level objects that are currently available in the current NETCONF session, or which have been loaded with the **mgrload** command.

The **list oids** command will display the object identifiers of the top-level objects that are currently available in the current NETCONF session, or which have been loaded with the **mgrload** command.

The **list scripts** command will display the script files that are in the current script search path. The module parameter has no affect in this mode.

#### list command

Command type:	local
Default parameter:	none
Min parameters:	1
Max parameters:	6

## Yuma Tools User Manual

Return type:	data
YANG file:	yangcli.yang

### Command Parameters:

- **choice help-mode** (not entered)  
This parameter is ignored if the listtype choice is set to ' the **history show** command is entered.
  - **brief**
    - type: empty
    - usage: optional
    - default: none
    - This parameter specifies that brief documentation mode should be used.
  - **normal**
    - type: empty
    - usage: optional
    - default: none
    - This parameter specifies that normal documentation mode should be used.
  - **full**
    - type: empty
    - usage: optional
    - default: none
    - This parameter specifies that full documentation mode should be used.
- **choice 'listtype'** (not entered)
  - **usage**: mandatory
  - **default**: none
  - This parameter specifies the what type of data should be listed.
  - **commands**
    - type: empty
    - usage: mandatory
    - default: none
    - This parameter specifies that the available commands should be listed.
      - If the **help-mode** is set to 'brief', then only the command names will be listed.
      - If the **help-mode** is set to 'normal', then the XML prefix and the command name will be listed.
      - If the **help-mode** is set to 'full', then the module name and the command name will be listed.
  - **files**
    - type: empty
    - usage: mandatory

## Yuma Tools User Manual

- default: none
- This parameter specifies that all the data files in the current data search path should be listed.
- **modules**
  - type: empty
  - usage: mandatory
  - default: none
  - This parameter specifies that all the YANG files in the current module search path should be listed.
- **objects**
  - type: empty
  - usage: mandatory
  - default: none
  - This parameter specifies that the available top-level objects should be listed.
    - If the **help-mode** is set to 'brief', then only the object names will be listed.
    - If the **help-mode** is set to 'normal', then the XML prefix and the object name will be listed.
    - If the **help-mode** is set to 'full', then the module name and the object name will be listed.
- **oids**
  - type: empty
  - usage: mandatory
  - default: none
  - This parameter specifies that the available top-level object identifiers should be listed.
    - The **help-mode** parameter has no effect
- **scripts**
  - type: empty
  - usage: mandatory
  - default: none
  - This parameter specifies that all the script files in the current script search path should be listed.
- **module**
  - type: string
  - usage: optional
  - default: none
  - This parameter specifies a module name. If present then only information for the specified YANG module will be displayed.

Positive Response:

- the requested information will be printed

Negative Response:

## Yuma Tools User Manual

- An error message will be printed if errors are detected.

Usage:

```
yangcli andy@myagent> list objects full module=test
```

```
test:instance1
test:instance2
test:leafref1
test:leafref2
test:test1
test:test2
test:test3
test:idtest
test:musttest
test:anyxml.1
test:binary.1
test:bits.1
test:boolean.1
test:empty.1
test:enumeration.1
test:identityref.1
test:instance-identifier.1
test:instance-identifier.2
test:int8.1
test:int16.1
test:int32.1
test:int64.1
test:leafref.1
test:leafref.2
test:string.1
test:uint8.1
test:uint16.1
test:uint32.1
test:uint64.1
test:dec64.1
test:dec64.2
test:dec64.3
test:union.1
test:container.1
test:container.2
test:list.1
test:choice.1
test>xpath.1
```

```
yangcli andy@myagent>
```

## 7.4.24 LOAD

The **load** command is used to load a YANG module into the server.

This command is only supported by the **netconfd** server.

The YANG files must be available in the module search path for the server.

Refer to the **netconfd** configuration section for more details on adding new YANG modules into the server.

After using this command, the **mgrload** command may also be needed to keep the current session synchronized with the server.

Use the **revision** parameter to load a specific revision of a module.

The server will return the revision date of the module that was loaded (or already present).

### load command

Command type:	remote
Default parameter:	module
Min parameters:	1
Max parameters:	3
Return type:	data
YANG file:	yuma-system.yang

Command Parameters:

- module
  - type: string (length 1 .. 63)
  - usage: mandatory
  - default: none
  - This parameter specifies the name of the module to load.
- revision
  - type: date string (YYYY-MM-DD)
  - usage: optional
  - default: none
  - This parameter specifies the revision date of the module to load.
- deviation:
  - type: leaf-list of deviation module names
  - usage: optional (0 or more instances)
  - default: none

- This parameter specifies a deviation module to load prior to loading the requested module.

Positive Response:

- the server returns <mod-revision>

Negative Response:

- An error message will be printed if errors are detected locally.
- An <rpc-error> message will be printed if the server detected an error.

Usage:

```
yangcli andy@myagent> load toaster revision=2009-06-23
```

```
RPC Data Reply 27 for session 10:
```

```
rpc-reply {  
    mod-revision 2009-06-23  
}
```

```
yangcli andy@myagent>
```

### 7.4.25 lock

The **lock** command is used to request a global lock on a NETCONF database. It is used, along with the **unlock** command, to obtain exclusive write access to the NETCONF server.

The scope of a lock command is the lifetime of the session that requested the lock. This means that if the session that owns the lock is dropped for any reason, all the locks it holds will be released immediately by the server.

The use of database locks is optional in NETCONF, but it must be implemented by every server. It is strongly suggested that locks be used if multiple managers are likely to log into the particular NETCONF server.

One or more locks may be needed to execute a transaction safely:

- If the **:writable-running** capability is supported, then a lock on the <running> database is needed. This database can be locked at any time.
- If the **:startup** capability is supported, then a lock on the <startup> database is needed. This database can be locked at any time.
- If the **:candidate** capability is supported, then a lock on the <candidate> database is needed. A lock on the <running> database is also needed.
  - The <candidate> database can only be locked if there are no active edits in it.
  - The **discard-changes** command may be needed to clear a <candidate> database that has been left edited by a session that terminated unexpectedly.
  - Note: There is no way in NETCONF to tell the difference between an actively edited <candidate> database and an 'abandoned' <candidate> database. The server will almost never clear the <candidate> database. It will only clear any locks held. Use the discard-changes command (for other session's edits) with extreme caution.

#### lock command

## Yuma Tools User Manual

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

Command Parameters:

- **target**
  - type: container with 1 of N choice of leafs
  - usage: mandatory
  - default: none
  - 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
    - **url**
      - type: yang:uri
      - capabilities needed: :url, and the scheme used in the URL must be specified in the :url capability 'schemes' parameter.
      - To enter this parameter, the interactive mode must be used. The shorthand mode (target=url) cannot be used, since this parameter contains a string.

Positive Response:

- the server returns <ok/>

Negative Response:

- An error message will be printed if errors are detected locally.
- An <rpc-error> message will be printed if the server detected an error.

## Yuma Tools User Manual

- If the <error-tag> is 'lock-denied' then the <error-info> will contain a <session-id> leaf. This identifies the session number of the current lock holder.

Usage:

```
yangcli andy@myagent> lock target=candidate
```

```
RPC OK Reply 29 for session 10:
```

```
yangcli andy@myagent>
```

### 7.4.26 MERGE

The **merge** command is a high-level <edit-config> operation. It is used to merge some new nodes into the default target database.

A target node is specified (in 1 of 2 ways), and then the data structure is filled in. Only mandatory nodes will be filled in unless the **\$\$optional** system variable is set to 'true'.

Refer to the **fill** command for more details on interactive mode data structure completion.

#### **merge command**

Command type:	remote
Default parameter:	target
Min parameters:	1
Max parameters:	5
Return type:	status
YANG file:	yangcli.yang
Capabilities needed:	:candidate or :writable-running

Command Parameters:

- **choice 'from'** (not entered)
  - type: choice of case 'varref' or case 'from-cli'
  - usage: mandatory
  - default: none
  - This parameter specifies the where **yangcli** should get the data from, for the merge operation. It is either a user variable or from interactive input at the command line.
    - **varref**
      - type: string
      - usage: mandatory
      - default: none
      - This parameter specifies the name of the user variable to use for the target of the merge operation. The parameter must exist (e.g., created with the **fill** command) or an error message will be printed.

- **case from-cli** (not entered)
  - **target**
    - type: string
    - usage: mandatory
    - default: none
    - This parameter specifies the database target node of the merge operation. This is an **ncx:schema-instance** string, so any instance identifier, or absolute path expression, or something in between, is accepted.
  - **optional**
    - type: empty
    - usage: optional
    - default: controlled by **\$\$optional** system variable
    - This parameter controls whether optional nodes within the target will be filled in. It can be used to override the **\$\$optional** system variable, when it is set to 'false'.
  - **value**
    - type: anyxml
    - usage: mandatory
    - default: none
    - This parameter specifies the value that should be used for the contents of the **target** parameter. If it is entered, then the interactive mode prompting for missing parameters will be skipped, if this parameter is complete (or all mandatory nodes present if the **\$\$optional** system variable is 'false'). For example, if the target is a leaf, then specifying this parameter will always cause the interactive prompt mode to be skipped.
- **timeout**
  - type: uint32 (0 = no timeout, otherwise the number of seconds to wait)
  - usage: optional
  - default: set to the **\$\$timeout** system variable, default 30 seconds
  - This parameter specifies the number of seconds to wait for a response from the server before giving up. The session will not be dropped if a remote command times out, but any late response will be dropped. A value of zero means no timeout should be used, and **yangcli** will wait forever for a response.

## System Variables:

- **\$\$default-operation**
  - The <default-operation> parameter for the <edit-config> operation will be derived from this variable.
- **\$\$error-option**
  - The <error-option> parameter for the <edit-config> operation will be derived from this variable
- **\$\$test-option**
  - The <test-option> parameter for the <edit-config> operation will be derived from this variable

Positive Response:

- the server returns <ok/>

Negative Response:

- An error message will be printed if errors are detected locally.
- An <rpc-error> message will be printed if the server detected an error.

Usage:

```
yangcli andy@myagent> merge \
    target=/nacm/rules/moduleRule[moduleName='netconf']\
    [allowedRights='read write']/allowedGroup \
    value=ncx:guest

(no user prompting; <edit-config> request sent right away)
```

```
RPC OK Reply 31 for session 10:
```

```
yangcli andy@myagent>
```

Reference:

- RFC 4741, section 7.2

### 7.4.27 MGRLOAD

The **mgrload** command is used to load a YANG module into **yangcli**.

The YANG files must be available in the module search path for **yangcli**.

#### **mgrload command**

Command type:	local
Default parameter:	module
Min parameters:	1
Max parameters:	3
Return type:	status
YANG file:	yangcli.yang

Command Parameters:

- module
  - type: string (length 1 .. 63)
  - usage: mandatory
  - default: none

- This parameter specifies the name of the module to load.
- revision
  - type: date string (YYYY-MM-DD)
  - usage: optional
  - default: none
  - This parameter specifies the revision date of the module to load.
- deviation:
  - type: leaf-list of deviation module names
  - usage: optional (0 or more instances)
  - default: none
  - This parameter specifies a deviation module to load prior to loading the requested module.

Positive Response:

- OK

Negative Response:

- An error message will be printed if errors are detected.

Usage:

```
yangcli> mgrload toaster
```

```
Load module toaster OK
```

```
yangcli>
```

### 7.4.28 no-op

The **no-op** command is used to test server message processing response times, by providing a baseline response time to do nothing except return <ok/>.

It can also be used as an application-level keep-alive to prevent proxy idle timeout or server idle timeout problems from occurring.

This command is only supported by the **netconfd** server. The server will simply respond 'OK', if the request is well-formed.

#### **no-op command**

Command type:	remote
Default parameter:	none
Min parameters:	0
Max parameters:	0
Return type:	status
YANG file:	yuma-system.yang

Positive Response:

- the server returns <ok/>

Negative Response:

- An error message will be printed if errors are detected locally.
- An <rpc-error> message will be printed if the server detected an error.

Usage:

```
yangcli andy@myagent> no-op
```

```
RPC OK Reply 31 for session 10:
```

```
yangcli andy@myagent>
```

### 7.4.29 pwd

The **pwd** command is used to print the current working directory.

#### pwd command

Command type:	local
Default parameter:	none
Min parameters:	0
Max parameters:	0
Return type:	status
YANG file:	yangcli.yang

Positive Response:

- the current working directory is printed to the log or STDOUT

Negative Response:

- An error message will be printed if errors are detected.

Usage:

```
yangcli> pwd
```

```
Current working directory is /home/andy
```

```
yangcli>
```

### 7.4.30 quit

The **quit** command is used to terminate the **yangcli** program.

## Yuma Tools User Manual

If a NETCONF session is in progress, it will be dropped without sending a **close-session** command first. This should be taken into account if the server reports dropped TCP connections as an error.

### quit command

Command type:	local
Default parameter:	none
Min parameters:	0
Max parameters:	0
Return type:	none
YANG file:	yangcli.yang

Positive Response:

- The program terminates; no response will be printed.

Negative Response:

- An error message will be printed if errors are detected.

Usage:

```
yangcli> quit
```

```
andy@myworkstation>
```

### 7.4.31 RECALL

The **recall** command is used to recall a previously entered command line from the command line history buffer.

A command is recalled by its command ID number. Use the **history show** command to see the contents of the command line history buffer.

### recall command

Command type:	local
Default parameter:	index
Min parameters:	1
Max parameters:	1
Return type:	data
YANG file:	yangcli.yang

Positive Response:

- The specified command line is recalled into the current command line.

Negative Response:

- An error message will be printed if errors are detected.

Usage:

```
yangcli> recall 7
```

```
yangcli>sget /system
```

### 7.4.32 RELEASE-LOCKS

The **release-locks** command is used to release all the locks that were previously granted with the get-locks command.

If the get-locks command was not used, then this command will fail with an error message that no locks are active to unlock.

#### **release-locks command**

Command type:	remote
Default parameter:	none
Min parameters:	0
Max parameters:	0
Return type:	status
YANG file:	yangcli.yang

Positive Response:

- The previously granted locks are released.

Negative Response:

- An error message will be printed if errors are detected locally.
- An <rpc-error> message will be printed if the server detected an error.

Usage: (showing 2 locks being released)

```
yangcli ady@myagent> release-locks
```

```
RPC OK Reply 8 for session 23:
```

```
RPC OK Reply 9 for session 23:
```

```
yangcli andy@myagent>
```

### 7.4.33 REPLACE

The **replace** command is a high-level <edit-config> operation. It is used to replace an existing subtree with some new nodes, in the default target database.

Only the subtree indicated by the **target** or **varref** parameter will be replaced.

## Yuma Tools User Manual

A target node is specified (in 1 of 2 ways), and then the data structure is filled in. Only mandatory nodes will be filled in unless the **\$\$optional** system variable is set to 'true'.

Refer to the **fill** command for more details on interactive mode data structure completion.

### replace command

Command type:	remote
Default parameter:	target
Min parameters:	1
Max parameters:	5
Return type:	status
YANG file:	yangcli.yang
Capabilities needed:	:candidate or :writable-running

Command Parameters:

- **choice 'from'** (not entered)
  - type: choice of case 'varref' or case 'from-cli'
  - usage: mandatory
  - default: none
  - This parameter specifies the where **yangcli** should get the data from, for the replace operation. It is either a user variable or from interactive input at the command line.
    - **varref**
      - type: string
      - usage: mandatory
      - default: none
      - This parameter specifies the name of the user variable to use for the target of the replace operation. The parameter must exist (e.g., created with the **fill** command) or an error message will be printed.
    - **case from-cli** (not entered)
      - **target**
        - type: string
        - usage: mandatory
        - default: none
        - This parameter specifies the database target node of the replace operation. This is an **ncx:schema-instance** string, so any instance identifier, or absolute path expression, or something in between, is accepted.
      - **optional**
        - type: empty
        - usage: optional
        - default: controlled by **\$\$optional** system variable

## Yuma Tools User Manual

- This parameter controls whether optional nodes within the target will be filled in. It can be used to override the **\$\$optional** system variable, when it is set to 'false'.
- **value**
  - type: anyxml
  - usage: mandatory
  - default: none
  - This parameter specifies the value that should be used for the contents of the **target** parameter. If it is entered, then the interactive mode prompting for missing parameters will be skipped, if this parameter is complete (or all mandatory nodes present if the **\$\$optional** system variable is 'false'). For example, if the target is a leaf, then specifying this parameter will always cause the interactive prompt mode to be skipped.
- **timeout**
  - type: uint32 (0 = no timeout, otherwise the number of seconds to wait)
  - usage: optional
  - default: set to the **\$\$timeout** system variable, default 30 seconds
  - This parameter specifies the number of seconds to wait for a response from the server before giving up. The session will not be dropped if a remote command times out, but any late response will be dropped. A value of zero means no timeout should be used, and **yangcli** will wait forever for a response.

System Variables:

- **\$\$default-operation**
  - The <default-operation> parameter for the <edit-config> operation will be derived from this variable.
- **\$\$error-option**
  - The <error-option> parameter for the <edit-config> operation will be derived from this variable
- **\$\$test-option**
  - The <test-option> parameter for the <edit-config> operation will be derived from this variable

Positive Response:

- the server returns <ok/>

Negative Response:

- An error message will be printed if errors are detected locally.
- An <rpc-error> message will be printed if the server detected an error.

Usage:

```
yangcli andy@myagent> replace \
    target=/nacm/rules/moduleRule[moduleName='yuma-system']\
    [allowedRights='exec']

(user prompted to fill in specified <moduleRule> element)
```

```
RPC OK Reply 31 for session 10:
```

```
yangcli andy@myagent>
```

Reference:

- RFC 4741, section 7.2

## 7.4.34 RESTART

The **restart** command is used to restart the NETCONF server.

This command is only supported by the **netconfd** server.

The default access control configuration for **netconfd** will not allow any user except the designated 'superuser' account to invoke this command. Refer to the **netconfd** user manual for details on configuring access control.

### restart command

Command type:	remote
Default parameter:	none
Min parameters:	0
Max parameters:	0
Return type:	status
YANG file:	yuma-system.yang

Positive Response:

- the server will drop all sessions and restart

Negative Response:

- An error message will be printed if errors are detected locally.
- An <rpc-error> message will be printed if the server detected an error.

Usage:

```
yangcli andy@myagent> restart
```

```
ses: session 10 shut by remote peer
```

```
yangcli>
```

## 7.4.35 RUN

The run command is used to run a **yangcli** script.

## Yuma Tools User Manual

Refer to the section on scripts for details on how to write a script.

The script name is the only mandatory parameter. There are 9 generic parameters (named P1 to P9) that can be used to pass string parameters to scripts. Within a script, these are referenced with local variables (\$1 to \$9).

The run command can be used within a script.

Scripts do not return any status or data at this time.

The run command can appear inside a script, starting a new run level. An error will occur if a loop occurs or too many nest levels are used. Up to 64 run levels are supported in **yangcli**.

### run command

Command type:	local
Default parameter:	script
Min parameters:	1
Max parameters:	10
Return type:	status
YANG file:	yangcli.yang

Command Parameters:

- **P1, P2, P3, P4, P5, P6, P7, P8, P9**
  - type: string
  - usage: optional
  - default: none
  - These parameters provide a generic string parameter passing mechanism for each script invocation, similar to unix shell scripts. Within the script, the parameters **\$1, \$2, \$3, \$4, \$5, \$6, \$7, \$8** and **\$9** will be non-NULL only if the corresponding '**Pn**' parameter was set in the script invocation.
- **script**
  - type: string
  - usage: mandatory
  - default: none
  - This parameter specifies the name of the script to be run. If a path specification is included, then it will be used. Otherwise the current script search path will be used to find the script.

System Variables:

- **\$\$runpath**
  - Controls where **yangcli** will look for files specified in the **script** parameter.

Positive Response:

- the specified script will be executed

Negative Response:

- An error message will be printed if errors are detected locally.

## Yuma Tools User Manual

- An <rpc-error> message will be printed if the server detected an error, if any remote commands are contained in the script.

Usage:

```
yangcli> run connect P1=yangrocks

runstack: Starting level 1 script /home/andy/scripts/connect

(commands and responses are printed as if they were typed)

runstack: Ending level 1 script /home/andy/scripts/connect
yangcli andy@myserver>
```

### 7.4.36 **SAVE**

The **save** command is used to save NETCONF database edits to non-volatile storage, on the server. It is a pseudo-command, mapped to other remote commands depending on which database target is supported by the server (<candidate> or <running>).

The **save** command usually maps to either the **commit** or the **copy-config** command. Refer to the section on NETCONF sessions for more details.

#### **save command**

Command type:	remote
Default parameter:	none
Min parameters:	0
Max parameters:	0
Return type:	none
YANG file:	yangcli.yang

Positive Response:

- The server returns one or two <ok/> responses.

Negative Response:

- An error message will be printed if errors are detected locally.
- An <rpc-error> message will be printed if the server detected an error.

Usage:

```
yangcli andy@myserver> save

RPC OK Reply 34 on session 10

yangcli andy@myserver>
```

**7.4.37 SET-LOG-LEVEL**

The **set-log-level** command is used to configure the server logging verbosity level. It is only supported by the **netconfd** server.

This operation is defined as **nacm:secure**, so only the system super-user can invoke this command by default.

**set-log-level command**

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

Command Parameters:

- **log-level**
  - type: enumeration (off, error, warn, info, debug, debug2, debug3)
  - This mandatory parameter specifies the desired server logging verbosity level.

Positive Response:

- the server returns <ok/>

Negative Response:

- An error message will be printed if errors are detected locally.
- An <rpc-error> message will be printed if the server detected an error.]

```
yangcli andy@myagent> set-log-level log-level=debug2
```

```
RPC OK Reply 29 for session 10:
```

```
yangcli andy@myagent>
```

**7.4.38 SET-MY-SESSION**

The **set-my-session** command is used to configure the session customization parameters on the server. It is only supported by the **netconfd** server.

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

## Yuma Tools User Manual

Since all parameters are optional, they need to be entered in the same command line as the command name. Otherwise the **\$\$optional** system variable needs to be set to 'true'. If not, then no parameters will be sent to the server, and no session parameters will be changed.

### **set-my-session command**

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

Command Parameters:

- **indent**
  - type: uint32 (range: 0 .. 9)
  - This parameter specifies the desired number of spaces to indent for each new XML nest level, for the session. If missing, then the indent amount is not changed.
- **linesize**
  - type: uint32 (range: 40 .. 1024)
  - This parameter specifies the desired line length for the session. If missing, then the current line size is not changed.
- **with-defaults**
  - type: enumeration (report-all, trim, explicit)
  - This parameter specifies the desired default with-defaults filtering behavior for the session. If missing, the current with-defaults behavior is not changed.

Positive Response:

- the server returns <ok/>

Negative Response:

- An error message will be printed if errors are detected locally.
- An <rpc-error> message will be printed if the server detected an error.]

```
yangcli andy@myagent> set-my-session \
    --linesize=64 --with-defaults=trim
```

```
RPC OK Reply 25 for session 10:
```

```
yangcli andy@myagent>
```

## 7.4.39 sget

The **sget** command is used to invoke a NETCONF <get> operation with a subtree filter.

A target node is specified (in 1 of 2 ways), and then the data structure is filled in. Only mandatory nodes will be filled in unless the **\$\$optional** system variable is set to 'true'. This step can be skipped completely with the **nofill** parameter.

Refer to the **fill** command for more details on interactive mode data structure completion.

### sget command

Command type:	remote
Default parameter:	target
Min parameters:	1
Max parameters:	5
Return type:	data
YANG file:	yangcli.yang
Capabilities needed:	none
Capabilities optional:	:with-defaults

Command Parameters:

- **choice 'from'** (not entered)
  - type: choice of case 'varref' or case 'from-cli'
  - usage: mandatory
  - default: none
  - This parameter specifies the where **yangcli** should get the data from, for the subtree filter target of the <get> operation. It is either a user variable or from interactive input at the command line.
    - **varref**
      - type: string
      - usage: mandatory
      - default: none
      - This parameter specifies the name of the user variable to use for the subtree filter target of the <get> operation. The parameter must exist (e.g., created with the **fill** command) or an error message will be printed.
    - **case from-cli** (not entered)
      - **target**
        - type: string
        - usage: mandatory
        - default: none

## Yuma Tools User Manual

- This parameter specifies the database target node of the <get> operation. This is an **ncx:schema-instance** string, so any instance identifier, or absolute path expression, or something in between, is accepted.
  - The escape command ('?s') can be used in parameter mode to skip a parameter completely.
  - Entering the empty string for a parameter will cause a subtree selection node to be created instead of a content match node
- **optional**
  - type: empty
  - usage: optional
  - default: controlled by **\$\$optional** system variable
  - This parameter controls whether optional nodes within the target will be filled in. It can be used to override the **\$\$optional** system variable, when it is set to 'false'.
- **value**
  - type: anyxml
  - usage: mandatory
  - default: none
  - This parameter specifies the value that should be used for the contents of the **target** parameter. If it is entered, then the interactive mode prompting for missing parameters will be skipped, if this parameter is complete (or all mandatory nodes present if the **\$\$optional** system variable is 'false'). For example, if the target is a leaf, then specifying this parameter will always cause the interactive prompt mode to be skipped.
- **nofill**
  - type: empty
  - usage: optional
  - default: none
  - This parameter specifies the that no interactive prompting to fill in the contents of the specified subtree filter target will be done.
    - This causes the entire selected subtree to be requested in the filter.
    - yangcli will attempt to figure out if there can be multiple instances of the requested subtree. If not, then **nofill** will be the default for that target parameter.
- **with-defaults**
  - type: enumeration (none report-all trim explicit)
  - usage: optional
  - default: none
  - capabilities needed: with-defaults
  - This parameter controls how nodes containing only default values are returned in the <rpc-reply>.

System Variables:

- **\$\$timeout**

## Yuma Tools User Manual

- The response timeout interval will be derived from this system variable.
- **\$\$with-defaults**
  - The <with-defaults> parameter for the <get> operation will be derived from this system variable.

Positive Response:

- the server returns <data>

Negative Response:

- An error message will be printed if errors are detected locally.
- An <rpc-error> message will be printed if the server detected an error.

Output:

- **data**
  - type: anyxml
  - This element will contain the requested data from the <running> database or non-config data structures.

Usage:

```
yangcli andy@myagent> sget system

Filling container /system:
RPC Data Reply 32 for session 10:

rpc-reply {
    data {
        system {
            sysName myserver.localdomain
            sysCurrentDateTime 2009-07-06T02:24:19Z
            sysBootDateTime 2009-07-05T19:22:28Z
        }
    }
}

yangcli andy@myagent>
```

Reference:

- RFC 4741, section 7.7

### 7.4.40 SGET-CONFIG

The **sget-config** command is used to invoke a NETCONF <get-config> operation with a subtree filter. A target node is specified (in 1 of 2 ways), and then the data structure is filled in. Only mandatory nodes will be filled in unless the **\$\$optional** system variable is set to 'true'. This step can be skipped completely with the **nofill** parameter.

Refer to the **fill** command for more details on interactive mode data structure completion.

**sget-config command**

Command type:	remote
Default parameter:	target
Min parameters:	2
Max parameters:	6
Return type:	data
YANG file:	yangcli.yang
Capabilities needed:	none
Capabilities optional:	:candidate :startup :url :with-defaults

Command Parameters:

- **choice 'from'** (not entered)
  - type: choice of case 'varref' or case 'from-cli'
  - usage: mandatory
  - default: none
  - This parameter specifies the where **yangcli** should get the data from, for the subtree filter target of the <get> operation. It is either a user variable or from interactive input at the command line.
- **varref**
  - type: string
  - usage: mandatory
  - default: none
  - This parameter specifies the name of the user variable to use for the subtree filter target of the <get> operation. The parameter must exist (e.g., created with the **fill** command) or an error message will be printed.
- **case from-cli** (not entered)
  - **target**
    - type: string
    - usage: mandatory
    - default: none
    - This parameter specifies the database target node of the <get> operation. This is an **ncx:schema-instance** string, so any instance identifier, or absolute path expression, or something in between, is accepted.
      - The escape command ('?s') can be used in parameter mode to skip a parameter completely.
      - Entering the empty string for a parameter will cause a subtree selection node to be created instead of a content match node

- **optional**
  - type: empty
  - usage: optional
  - default: controlled by **\$\$optional** system variable
  - This parameter controls whether optional nodes within the target will be filled in. It can be used to override the **\$\$optional** system variable, when it is set to 'false'.
- **value**
  - type: anyxml
  - usage: mandatory
  - default: none
  - This parameter specifies the value that should be used for the contents of the **target** parameter. If it is entered, then the interactive mode prompting for missing parameters will be skipped, if this parameter is complete (or all mandatory nodes present if the **\$\$optional** system variable is 'false'). For example, if the target is a leaf, then specifying this parameter will always cause the interactive prompt mode to be skipped.
- **nofill**
  - type: empty
  - usage: optional
  - default: none
  - This parameter specifies the that no interactive prompting to fill in the contents of the specified subtree filter target will be done.
    - This causes the entire selected subtree to be requested in the filter.
    - yangcli will attempt to figure out if there can be multiple instances of the requested subtree. If not, then **nofill** will be the default for that target parameter.
- **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 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

## Yuma Tools User Manual

- **url**
  - type: yang:uri
  - capabilities needed: :url, and the scheme used in the URL must be specified in the :url capability 'schemes' parameter
  - To enter this parameter, the interactive mode must be used. The shorthand mode (source=url) cannot be used, since this parameter contains a string.
- **with-defaults**
  - type: enumeration (none report-all trim explicit)
  - usage: optional
  - default: none
  - capabilities needed: with-defaults
  - This parameter controls how nodes containing only default values are returned in the <rpc-reply>.

System Variables:

- **\$\$timeout**
  - The response timeout interval will be derived from this system variable.
- **\$\$with-defaults**
  - The <with-defaults> parameter for the <get-config> operation will be derived from this system variable.

Positive Response:

- the server returns <data>

Negative Response:

- An error message will be printed if errors are detected locally.
- An <rpc-error> message will be printed if the server detected an error.

Output:

- **data**
  - type: anyxml
  - This element will contain the requested data from the database indicated by the **source** parameter.

Usage:

```
yangcli andy@myagent> sget-config /nacm/rules/dataRule \
    source=candidate \
    nofill
```

RPC Data Reply 35 for session 10:

```
rpc-reply {
    data {
        nacm {
            rules {
```

## Yuma Tools User Manual

```
dataRule nacm-tree {  
    name nacm-tree  
    path /nacm:nacm  
    allowedRights {  
        read  
        write  
    }  
    allowedGroup nacm:admin  
    comment 'access to nacm config'  
}  
}  
}  
}  
}  
}  
  
yangcli andy@myagent>
```

### Reference:

- RFC 4741, section 7.1

## 7.4.41 show

The **show** command is used to show program status and YANG details that may be needed during management of a NETCONF server.

There are several variants of the **show** command:

- The **show global** command prints the contents of one global user variable.
- The **show globals** command prints a summary or the contents of all the global user variables.
- The **show local** command prints the contents of one local user variable.
- The **show locals** command prints a summary or the contents of all the local user variables.
- The **show module** command prints meta information or help text for one YANG module.
- The **show modules** command prints meta information for all the currently available YANG modules. IF a session is active, this will be the list of modules the server reported, plus any modules loaded with the **mgrload** command.
- The **show objects** command prints the available objects or help text for the available objects.
- The **show var** command prints the contents of the specified variable.
- The **show vars** command prints a summary or the contents of all the program variables.
- The **show version** command prints the **yangcli** version number

### show command

Command type:	local
Default parameter:	none

## Yuma Tools User Manual

Min parameters:	1
Max parameters:	2
Return type:	data
YANG file:	yangcli.yang

### Command Parameters:

- **choice help-mode** (not entered) (default choice is 'normal')
  - **brief**
    - type: empty
    - usage: optional
    - default: none
    - This parameter specifies that brief documentation mode should be used.
  - **normal**
    - type: empty
    - usage: optional
    - default: none
    - This parameter specifies that normal documentation mode should be used.
  - **full**
    - type: empty
    - usage: optional
    - default: none
    - This parameter specifies that full documentation mode should be used.
- **choice showtype** (not entered)
  - mandatory 1 of N choice for the sub-command for the **show** command
    - **global**
      - type: string
      - usage: mandatory
      - default: none
      - This parameter specifies the name of the global variable to show information for.
        - If **help-mode** is 'brief', then the name, variable object, and type of the global variable object will be printed.
        - If **help-mode** is 'normal' or 'full', then the contents of the specified global variable will be printed.
    - **globals**
      - type: empty
      - usage: mandatory
      - default: none
      - This parameter specifies that information for all global variables should be printed.

## Yuma Tools User Manual

- If **help-mode** is 'brief', then the name, variable object, and type of each global variable object will be printed.
  - If **help-mode** is 'normal' or 'full', then the contents of each global variable will be printed.
- **local**
    - type: string
    - usage: mandatory
    - default: none
    - This parameter specifies the name of the local variable to show information for.
      - If **help-mode** is 'brief', then the name, variable object, and type of the local variable object will be printed.
      - If **help-mode** is 'normal' or 'full', then the contents of the specified local variable will be printed.
  - **locals**
    - type: empty
    - usage: mandatory
    - default: none
    - This parameter specifies that information for all local variables should be printed.
      - If **help-mode** is 'brief', then the name, variable object, and type of each local variable object will be printed.
      - If **help-mode** is 'normal' or 'full', then the contents of each local variable will be printed.
  - **module**
    - type: string
    - usage: mandatory
    - default: none
    - This parameter specifies the name of the module to show information for.
      - If **help-mode** is 'brief' or 'normal', then the name, version, namespace, and source of the module will be printed.
      - If **help-mode** is 'full', then the module level help text for the specified module will be printed.
  - **modules**
    - type: empty
    - usage: mandatory
    - default: none
    - This parameter specifies that information for all available modules should be printed:
      - If **help-mode** is 'brief', then the name of each module will be printed.
      - If **help-mode** is 'normal', then the XML prefix, name, and revision date of each module will be printed.
      - If **help-mode** is 'full', then the name, revision date, prefix, namespace, and source of each module will be printed.

# Yuma Tools User Manual

- **objects**
  - type: empty
  - usage: mandatory
  - default: none
  - This parameter specifies that information for all available database objects should be printed:
    - If **help-mode** is 'brief', then the module name and name of each object will be printed.
    - If **help-mode** is 'normal', then the YANG object type, object name, and YANG base type will be printed.
  - If **help-mode** is 'full', then brief help text will be printed for every object.
- **var**
  - type: string
  - usage: mandatory
  - default: none
  - This parameter specifies the name of any variable to show information for.
    - If **help-mode** is 'brief', then the name, variable object, and type of the variable object will be printed.
    - If **help-mode** is 'normal' or 'full', then the contents of the specified variable will be printed.
- **vars**
  - type: empty
  - usage: mandatory
  - default: none
  - This parameter specifies that information for all variables should be printed. This includes all CLI, read-only system, read-write system, global user, and local user variables.
    - If **help-mode** is 'brief', then the name, variable object, and type of each variable object will be printed.
    - If **help-mode** is 'normal' or 'full', then the contents of each variable will be printed.
- **version**
  - type: empty
  - usage: mandatory
  - default: none
  - This parameter specifies that the yangcli program version string should be printed.
    - The **help-mode** parameter has no affect in this mode.

Positive Response:

- the server prints the requested information

Negative Response:

- An error message will be printed if errors are detected.

## Yuma Tools User Manual

Usage:

```
yangcli andy@myagent> show modules

nc:netconf/2009-04-10
inet:ietf-inet-types/2009-05-13
ns:ietf-netconf-state/2009-03-03
wd:ietf-with-defaults/2009-04-10
yang:ietf-yang-types/2009-05-13
nacm:nacm/2009-05-13
manageEvent:nc-notifications/2008-07-14
ncx:ncx/2009-06-12
ncxapp:ncx-app-common/2009-04-10
nt:ncxtypes/2008-07-20
nd:netconfd/2009-05-28
ncEvent:notifications/2008-07-14
sys:system/2009-06-04
t:test/2009-06-10

yangcli andy@myagent>
```

### 7.4.42 SHUTDOWN

The **shutdown** command is used to shut down the NETCONF server.

This command is only supported by the **netconfd** server.

The default access control configuration for **netconfd** will not allow any user except the designated 'superuser' account to invoke this command. Refer to the **netconfd** user manual for details on configuring access control.

#### shutdown command

Command type:	remote
Default parameter:	none
Min parameters:	0
Max parameters:	0
Return type:	status
YANG file:	yuma-system.yang

Positive Response:

- the server will drop all sessions and terminate.

Negative Response:

- An error message will be printed if errors are detected locally.
- An <rpc-error> message will be printed if the server detected an error.

Usage:

```
yangcli andy@myagent> shutdown
ses: session 10 shut by remote peer
yangcli>
```

## 7.4.43 UNLOCK

The **unlock** command is used to request a global lock on a NETCONF database. It is used, along with the **lock** command, to obtain exclusive write access to the NETCONF server.

### unlock command

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

Command Parameters:

- **target**
  - type: container with 1 of N choice of leafs
  - usage: mandatory
  - default: none
  - 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
- **url**
  - type: yang:uri
  - capabilities needed: :url, and the scheme used in the URL must be specified in the :url capability 'schemes' parameter.
  - To enter this parameter, the interactive mode must be used. The shorthand mode (target=url) cannot be used, since this parameter contains a string.

Positive Response:

- the server returns <ok/>

Negative Response:

- An error message will be printed if errors are detected locally.
- An <rpc-error> message will be printed if the server detected an error.

Usage:

```
yangcli andy@myagent> unlock target=candidate
```

```
RPC OK Reply 38 for session 10:
```

```
yangcli andy@myagent>
```

### 7.4.44 VALIDATE

The **validate** command is used to check if a complete NETCONF database is valid or not.

The **:validate** capability must be supported by the server to use this command.

#### **validate command**

Command type:	remote
Default parameter:	none
Min parameters:	1
Max parameters:	1
Return type:	status
YANG file:	yuma-netconf.yang
Capabilities needed:	:validate
Capabilities optional:	:candidate :startup :url

Command Parameters:

- **target**
  - type: container with 1 of N choice of leafs

- usage: mandatory
- default: none
- This parameter specifies the name of the target database to 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
  - **url**
    - type: yang:uri
    - capabilities needed: :url, and the scheme used in the URL must be specified in the :url capability 'schemes' parameter.
    - To enter this parameter, the interactive mode must be used. The shorthand mode (target=url) cannot be used, since this parameter contains a string.
  - **config**
    - type: anyxml
    - capabilities
    - This parameter represents an entire database, using the in-line config form, similar to the **edit-config** command, except this config parameter contains no nc:operation attributes and must be a complete database, not a subset.

Positive Response:

- the server returns <ok/>

Negative Response:

- An error message will be printed if errors are detected locally.
- An <rpc-error> message will be printed if the server detected an error.

Usage:

```
yangcli andy@myagent> validate source=candidate
```

```
RPC OK Reply 36 for session 10:
```

```
yangcli andy@myagent>
```

### 7.4.45 xGET

## Yuma Tools User Manual

The **xget** command is used to invoke a NETCONF <get> operation with an XPath filter.

### xget command

Command type:	remote
Default parameter:	select
Min parameters:	1
Max parameters:	3
Return type:	data
YANG file:	yangcli.yang
Capabilities needed:	none
Capabilities optional:	:with-defaults

Command Parameters:

- **choice 'from'** (not entered)
  - type: choice of case 'varref' or case 'select'
  - usage: mandatory
  - default: none
  - This parameter specifies the where **yangcli** should get the data from, for the XPath filter target of the <get> operation. It is an XPath expression, either contained in a user variable or directly from the **select** parameter.
    - **varref**
      - type: string
      - usage: mandatory
      - default: none
      - This parameter specifies the name of the user variable the contains the XPath expression for the XPath filter in the <get> operation. The parameter must exist (e.g., created with an assignment statement) or an error message will be printed.
    - **select**
      - type: string
      - usage: mandatory
      - default: none
      - This parameter specifies the XPath expression to use for the XPath filter in the <get> operation.
- **timeout**
  - type: uint32 (0 = no timeout, otherwise the number of seconds to wait)
  - usage: optional
  - default: set to the **\$\$timeout** system variable, default 30 seconds
  - This parameter specifies the number of seconds to wait for a response from the server before giving up. The session will not be dropped if a remote command times out, but any

## Yuma Tools User Manual

late response will be dropped. A value of zero means no timeout should be used, and **yangcli** will wait forever for a response.

- **with-defaults**

- type: enumeration (none report-all trim explicit)
- usage: optional
- default: none
- capabilities needed: with-defaults
- This parameter controls how nodes containing only default values are returned in the <rpc-reply>.

System Variables:

- **\$\$timeout**

- The response timeout interval will be derived from this system variable.

- **\$\$with-defaults**

- The <with-defaults> parameter for the <get> operation will be derived from this system variable.

Positive Response:

- the server returns <data>

Negative Response:

- An error message will be printed if errors are detected locally.
- An <rpc-error> message will be printed if the server detected an error.

Output:

- **data**

- type: anyxml
- This element will contain the requested data from the <running> database or non-config data structures.

Usage:

```
yangcli andy@myagent> xget //name
```

```
RPC Data Reply 42 for session 10:
```

```
rpc-reply {
    data {
        nacm {
            rules {
                dataRule nacm-tree {
                    name nacm-tree
                }
            }
        }
    }
    xpath.1 {
        name barney
    }
}
```

```

}
ietf-netconf-state {
    datastores {
        datastore {
            name {
                candidate
            }
        }
    }
}

netconf {
    streams {
        stream NETCONF {
            name NETCONF
        }
    }
}
}
}
}

yangcli andy@myagent>

```

Reference:

- RFC 4741, section 7.7

## 7.4.46 xget-config

The **xget-config** command is used to invoke a NETCONF <get-config> operation with an XPath filter.

### **xget-config command**

Command type:	remote
Default parameter:	select
Min parameters:	2
Max parameters:	4
Return type:	data
YANG file:	yangcli.yang
Capabilities needed:	none
Capabilities optional:	:candidate :startup :url

	:with-defaults
--	----------------

## Command Parameters:

- **choice 'from'** (not entered)
  - type: choice of case 'varref' or case 'select'
  - usage: mandatory
  - default: none
  - This parameter specifies the where **yangcli** should get the data from, for the XPath filter target of the <get-config> operation. It is an XPath expression, either contained in a user variable or directly from the **select** parameter.
- **varref**
  - type: string
  - usage: mandatory
  - default: none
  - This parameter specifies the name of the user variable the contains the XPath expression for the XPath filter in the <get-config> operation. The parameter must exist (e.g., created with an assignment statement) or an error message will be printed.
- **select**
  - type: string
  - usage: mandatory
  - default: none
  - This parameter specifies the XPath expression to use for the XPath filter in the <get-config> operation.
- **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 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
    - **url**

## Yuma Tools User Manual

- type: yang:uri
- capabilities needed: :url, and the scheme used in the URL must be specified in the :url capability 'schemes' parameter
- To enter this parameter, the interactive mode must be used. The shorthand mode (source=url) cannot be used, since this parameter contains a string.
- **timeout**
  - type: uint32 (0 = no timeout, otherwise the number of seconds to wait)
  - usage: optional
  - default: set to the **\$\$timeout** system variable, default 30 seconds
  - This parameter specifies the number of seconds to wait for a response from the server before giving up. The session will not be dropped if a remote command times out, but any late response will be dropped. A value of zero means no timeout should be used, and **yangcli** will wait forever for a response.
- **with-defaults**
  - type: enumeration (none report-all trim explicit)
  - usage: optional
  - default: none
  - capabilities needed: with-defaults
  - This parameter controls how nodes containing only default values are returned in the <rpc-reply>.

System Variables:

- **\$\$timeout**
  - The response timeout interval will be derived from this system variable.
- **\$\$with-defaults**
  - The <with-defaults> parameter for the <get-config> operation will be derived from this system variable.

Positive Response:

- the server returns <data>

Negative Response:

- An error message will be printed if errors are detected locally.
- An <rpc-error> message will be printed if the server detected an error.

Output:

- **data**
  - type: anyxml
  - This element will contain the requested data from the **source** database.

Usage:

```
yangcli andy@myagent> xget /nacm/groups \
    source=running
```

RPC Data Reply 41 for session 10:

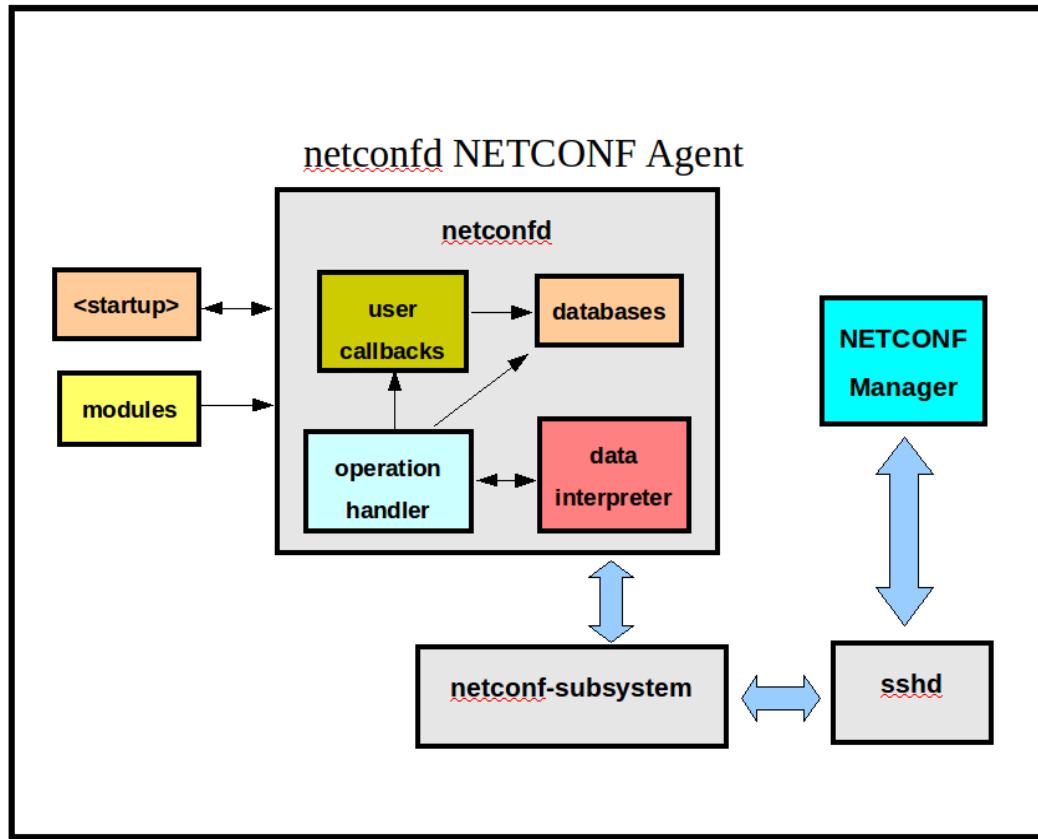
## Yuma Tools User Manual

```
rpc-reply {  
    data {  
        nacm {  
            groups {  
                group nacm:admin {  
                    groupIdentity nacm:admin  
                    userName andy  
                    userName fred  
                    userName barney  
                }  
                group nacm:guest {  
                    groupIdentity nacm:guest  
                    userName guest  
                }  
            }  
        }  
    }  
}
```

yangcli andy@svnserver>

# 8 netconfd User Guide

## netconfd Program Components



## 8.1 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.

### 8.1.1 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

## Yuma Tools User Manual

- Supports all of RFC 4741, except the optional **:url** capability, which will be added in the near future:
- 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**.
- 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-state** 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.

## 8.1.2 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.

## 8.1.3 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.

## 8.1.4 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**.

## Yuma Tools User Manual

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/netconfd.conf
```

### 8.1.5 STOPPING NETCONFD

## Yuma Tools User Manual

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.

### 8.1.6 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.

### 8.1.7 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

## Yuma Tools User Manual

- 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.

### 8.1.8 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-state	standard NETCONF monitoring, and the <get-schema> operation
ietf-with-defaults	<with-defaults> extension
ietf-yang-types	standard data types
interfaces	network interfaces information
nacm	NETCONF Access Control Model

nc-notifications	standard replay notifications
ncx	Netconf Central extensions
ncx-app-common	Common CLI parameters
ncxtypes	Netconf Central data types
netconfd	CLI parameters and extra operations
notifications	standard notification operations
proc	/proc file system monitoring
system	system monitoring, operations, and notifications
test	test module in DEBUG builds

## 8.1.9 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
system	<sysStartup>	server startup event
system	<sysSessionStart>	NETCONF session started
system	<sysSessionEnd>	NETCONF session ended
system	<sysConfigChange>	<running> configuration has changed
system	<sysCapabilityChange>	server capability added or deleted
system	<sysConfirmedCommit>	confirmed-commit procedure event

## 8.1.10 OPERATION SUMMARY

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

### Pre-loaded Operations

module	operation	description
netconf	<close-session>	Terminate the current session.
netconf	<commit>	Activate edits in <candidate>.
netconf	<copy-config>	Copy an entire configuration.

## Yuma Tools User Manual

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

### 8.1.11 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	Specifies one or more YANG modules to load as deviations
--eventlog-size	Specifies the maximum number of events stored in the notification replay buffer.

## Yuma Tools User Manual

--help	Get context-sensitive help
--help-mode	Adjust the help output (--brief or --full)
--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
--port	Specifies up to 4 TCP port numbers to accept NETCONF connections from
--runpath	Script search path (for instrumentation use only at this time)
--start	Specifies the startup configuration file location with --startup to override the default. Also, --no-startup can be used to skip the startup load completely.
--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-validate	Enable or disable the :validate capability
--yuma-home	Specifies the <b>\$YUMA_HOME</b> project root to use when searching for files

## 8.2 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.

### 8.2.1 :CANDIDATE

## Yuma Tools User Manual

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.

### 8.2.2 :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.

### 8.2.3 :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.

### 8.2.4 :NETCONF-MONITORING

The **:netconf-monitoring** capability indicates that the /ietf-netconf-state 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 /ietf-netconf-state/capabilities subtree can be examined to discover the active set of NETCONF capabilities.

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

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

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

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

### **8.2.5 :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.

### **8.2.6 :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.

### **8.2.7 :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.

### **8.2.8 :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>.

### 8.2.9 :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.

### 8.2.10 :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 their 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.

## 8.2.11 :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'.

## 8.2.12 :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.

## 8.3 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.

## Yuma Tools User Manual

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.

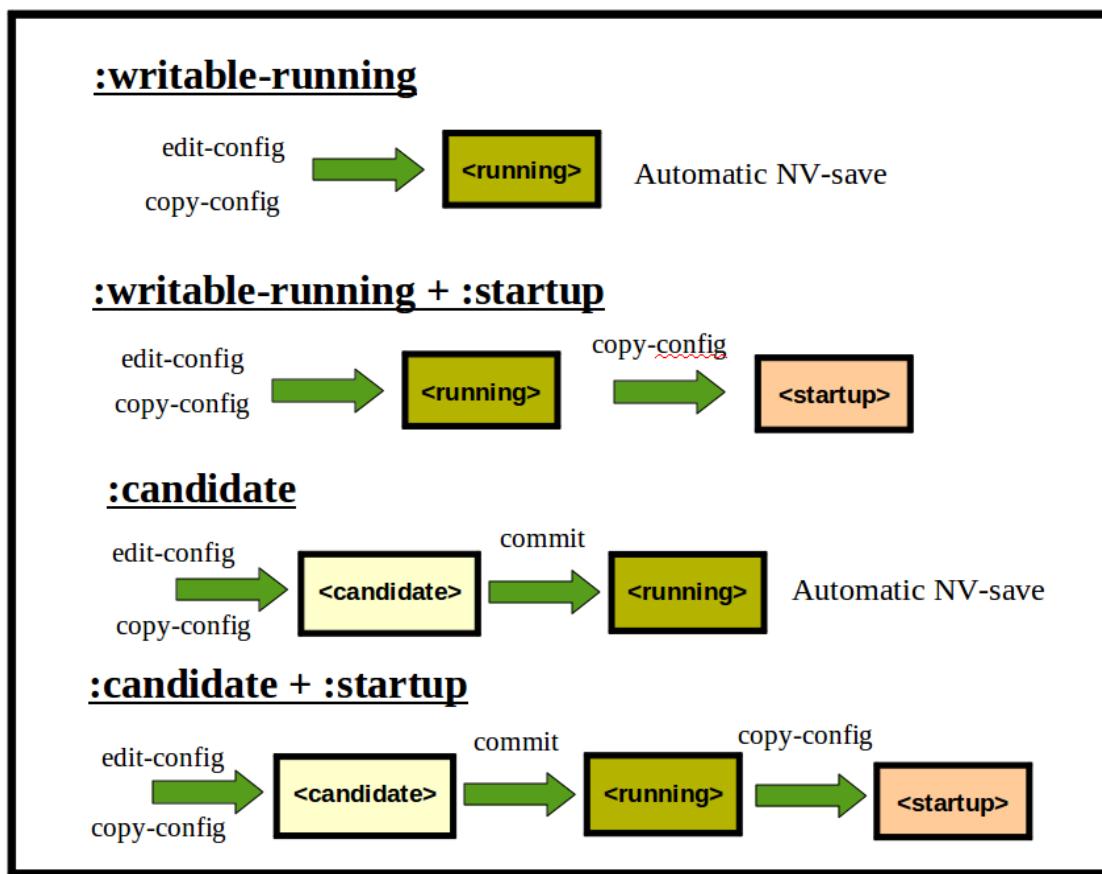
The exact databases that are present in the server are controlled by 3 capabilities:

- :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:



### 8.3.1 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.

## Yuma Tools User Manual

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.

### 8.3.2 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.

### 8.3.3 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.

### 8.3.4 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.

## 8.4 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.

### 8.4.1 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-zA-Z][a-zA-Z,0-9]{0,62}';  
    }  
}
```

- A user name can 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'].

### 8.4.2 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.

### 8.4.3 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.

## Yuma Tools User Manual

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>
```

## Yuma Tools User Manual

```
<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&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&revision=2009-05-13
</nc:capability>
<nc:capability>
    urn:ietf:params:xml:ns:netconf:state?module=ietf-netconf-state&revision=2009-06-
16
</nc:capability>
<nc:capability>
    urn:ietf:params:netconf:capability:with-defaults:1.0?module=ietf-with-
defaults&revision=2009-07-01&features=with-defaults
</nc:capability>
<nc:capability>
    urn:ietf:params:xml:ns:yang:yang-types?module=ietf-yang-types&revision=2009-05-13
</nc:capability>
<nc:capability>
    http://netconfcentral.com/ns/interfaces?module=interfaces&rQ

ses_msg: setup send buff 1
evision=2009-07-17
</nc:capability>
<nc:capability>
    http://netconfcentral.com/ns/mysession?module=mysession&revision=2009-08-11
</nc:capability>
<nc:capability>
    http://netconfcentral.com/ns/nacm?module=nacm&revision=2009-05-13
</nc:capability>
<nc:capability>
```

## Yuma Tools User Manual

```
urn:ietf:params:xml:ns:netmod:notification?module=nc-notifications&revision=2008-07-14
</nc:capability>
<nc:capability>
    http://netconfcentral.com/ncx?module=ncx&revision=2009-06-12
</nc:capability>
<nc:capability>
    http://netconfcentral.com/ns/ncx-app-common?module=ncx-app-common&revision=2009-04-10
</nc:capability>
<nc:capability>
    http://netconfcentral.com/ns/ncxtypes?module=ncxtypes&revision=2008-07-20
</nc:capability>
<nc:capability>
    http://netconfcentral.com/ns/netconfd?module=netconfd&revision=2009-05-28
</nc:capability>
<nc:capability>
    urn:ietf:params:xml:ns:netconf:notification:1.0?
module=notifications&revision=2008-07-14
</nc:capability>
<nc:capability>
    http://netconfcentral.com/ns/proc?module=proc&revision=2009-07-17
</nc:capability>
<nc:capability>
    http://netconfcentral.com/ns/system?module=system&revision=2009-06-04
</nc:capability>
<nc:capability>
    http://netconfcentral.com/ns/test?module=test&revision=2009-06-10&features=feature1,feature3,feature4
</nc:capability>
</nc:capabilities>
<nc:session-id>1</nc:session-id>
</nc:hello>]]>]]>
```

### 8.4.4 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>
```

### 8.4.5 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 with 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.

### 8.4.6 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 **/ietf-netconf-state/sessions** container
- will generate a <sysSessionEnd> notification entry for the closed or killed session

## 8.5 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.

### 8.5.1 <error-severity> Element

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

### 8.5.2 <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

<b>error-tag</b>	<b>error-info</b>	<b>description</b>
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

### 8.5.3 <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 occurred, 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**

<b>error-app-tag</b>	<b>description</b>
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)

**8.5.4 <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>

```

## 8.5.5 <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.

## 8.5.6 <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

## 8.5.7 INSTANCE-REQUIRED ERROR EXAMPLE

## Yuma Tools User Manual

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**

<b>data</b>	<b>description</b>
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.com/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.com/ns/test"
    xmlns:ncx="http://netconfcentral.com/ncx">
    <nc:rpc-error>
        <nc:error-type>application</nc:error-type>
    </nc:rpc-error>
</nc:rpc-reply>
```

```

<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>

```

## 8.5.8 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

<b>data</b>	<b>description</b>
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.com/ns/test"/>
        </nc:config>
    </nc:edit-config>
</nc:rpc>

```

```
</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.com/ns/test"
  xmlns:ncx="http://netconfcentral.com/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>
```

### 8.5.9 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

<b>data</b>	<b>description</b>
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"?>
```

## Yuma Tools User Manual

```
<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.com/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>
```

### 8.5.10 NOT-IN-RANGE ERROR EXAMPLE

The not-in-range error is generated when a numeric type violates 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.com/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.com/ns/test">1000</t:int8.1>
    </nc:config>
  </nc:edit-config>
</nc:rpc>
```

## 8.6 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,

### **8.6.1 <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>
```

## 8.6.2 <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.

If 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>
```

## 8.6.3 <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

## Yuma Tools User Manual

	: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 allow the superuser account 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**

## Yuma Tools User Manual

- 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>
```

### 8.6.4 <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.

## Yuma Tools User Manual

The subscription will continue until the session is closed, unless the <stopTime> parameter is present. In that case, the subscription will terminate at that time (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 to store the most recent 1000 system notification events.

An entry will be created in the 'subscriptions' data structure in the ietf-netconf-state 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

## Yuma Tools User Manual

- 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>
```

### 8.6.5 <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.

## Yuma Tools User Manual

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>
```

## 8.6.6 <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>
```

## 8.6.7 <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-set' 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.com/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>
```

## 8.6.8 <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'**

## Yuma Tools User Manual

- 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.com/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"?>
```

## Yuma Tools User Manual

```
<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.com/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>
```

### 8.6.9 <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**

## Yuma Tools User Manual

- 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">
```

## Yuma Tools User Manual

```
<nacm:nacm xmlns:nacm="http://netconfcentral.com/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.com/ns/nacm">
  <nacm:rules>
    <nacm:moduleRule>
      <nacm:moduleName>netconf</nacm:moduleName>
      <nacm:allowedRights>read exec</nacm:allowedRights>
      <nacm:allowedGroup>nacm:guest</nacm:allowedGroup>
    </nacm:moduleRule>
    <nacm:moduleRule>
      <nacm:moduleName>netconfd</nacm:moduleName>
      <nacm:allowedRights>read write exec</nacm:allowedRights>
      <nacm:allowedGroup>nacm:admin</nacm:allowedGroup>
      <nacm:comment>access to shutdown and restart rpcs</nacm:comment>
    </nacm:moduleRule>
    <nacm:moduleRule>
```

## Yuma Tools User Manual

```
<nacm:moduleName>netconf</nacm:moduleName>
<nacm:allowedRights>read write exec</nacm:allowedRights>
<nacm:allowedGroup>nacm:admin</nacm:allowedGroup>
<nacm:allowedGroup>nacm:monitor</nacm:allowedGroup>
</nacm:moduleRule>
</nacm:rules>
</nacm:nacm>
</nc:data>
</nc:rpc-reply>
```

### 8.6.10 <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.com/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.com/ns/mysession">
        3
    </myses:indent>
    <myses:linesize xmlns:myses="http://netconfcentral.com/ns/mysession">
        72
    </myses:linesize>
    <myses:with-defaults xmlns:myses="http://netconfcentral.com/ns/mysession">
        report-all
    </myses:with-defaults>
</nc:rpc-reply>
```

## **8.6.11 <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:

## Yuma Tools User Manual

- **identifier**
  - type: string
  - This parameter specifies the name of the module to retrieve.
    - Do not use any path specification or 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>
```

## 8.6.12 <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>
```

## 8.6.13 <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

## Yuma Tools User Manual

- 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.com/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.com/ns/netconfd">
    2008-10-15
  </nd:mod-revision>
</nc:rpc-reply>
```

### 8.6.14 <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

## Yuma Tools User Manual

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>
```

## 8.6.15 <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.com/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>
```

## 8.6.16 <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.com/ns/netconfd"/>
</nc:rpc>
```

Example Reply:

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

## 8.6.17 <SET-LOG-LEVEL>

## Yuma Tools User Manual

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.com/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>
```

## 8.6.18 <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"
```

## Yuma Tools User Manual

```
message-id="3">
<myses:set-my-session xmlns:myses="http://netconfcentral.com/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>
```

### 8.6.19 <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.com/ns/netconfd"/>
</nc:rpc>
```

Example Reply:

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

## 8.6.20 <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>
```

### 8.6.21 <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

## Yuma Tools User Manual

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>
```

## 8.7 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

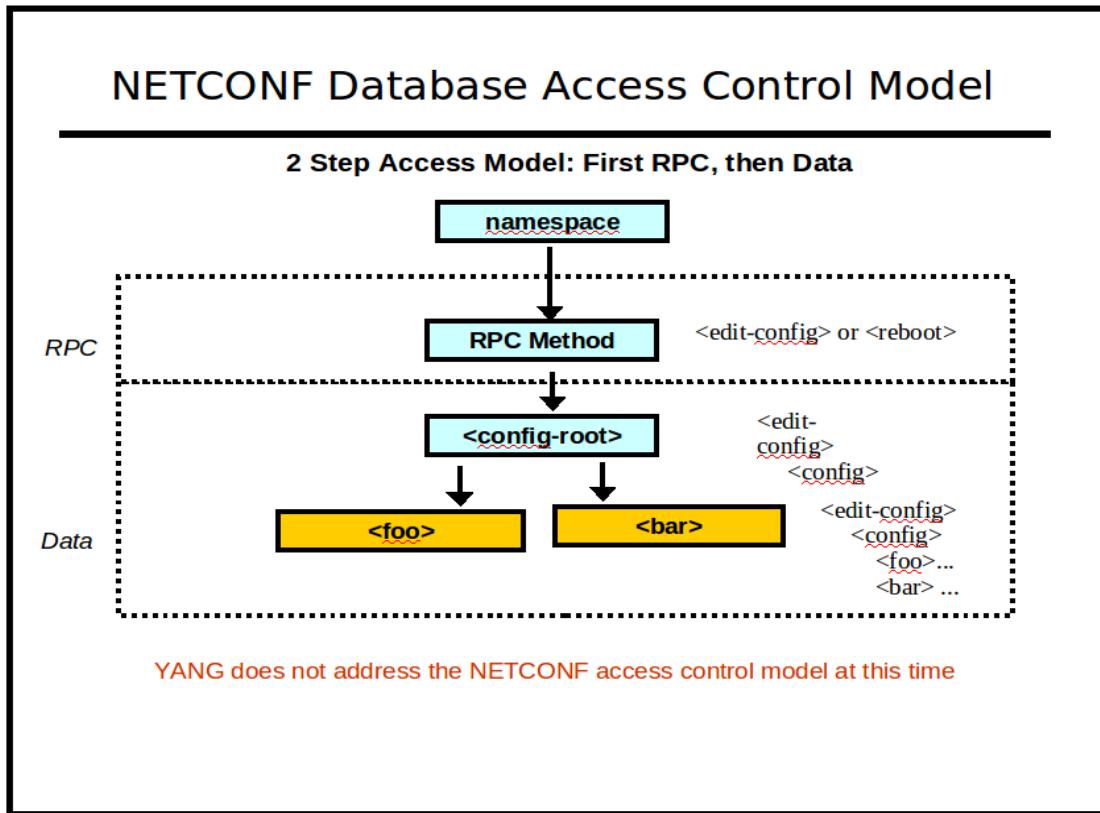
Rules apply to one or more groups.

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.

### 8.7.1 NACM MODULE STRUCTURE

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

- leaf **<noRuleReadDefault>**: permit or deny read access when no rule found
- leaf **<noRuleWriteDefault>**: permit or deny write access when no rule found
- leaf **<noRuleExecDefault>**: permit or deny execute access when no rule found
- 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 **<groupIdentity>**: group identifier string
    - leaf-list **<userName>**: leaf-list of users that belong to the group
- container **<rules>**:
  - list **<moduleRule>**: an access rule for an entire module namespace

- leaf <**moduleName**>: [key] name of the YANG module associated with the rule
- leaf <**allowedRights**>: [key] privileges granted to all groups associated with the rule
- leaf-list <**allowedGroup**>: 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 <**rpcRule**>: an access rule for one specific protocol operation, defined in a YANG rpc statement.
  - leaf <**rpcModuleName**>: [key] name of the YANG module associated with the rule
  - leaf <**rpcName**>: [key] name of the RPC operation associated with the rule
  - leaf <**allowedRights**>: [key] privileges granted to all groups associated with the rule
  - leaf-list <**allowedGroup**>: 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 <**dataRule**>: 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 <**allowedRights**>: privileges granted to all groups associated with the rule
  - leaf-list <**allowedGroup**>: 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)

### 8.7.2 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.

### 8.7.3 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 <allowedGroup> 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.com/ns/nacm">  
                <nacm:groups>  
                    <nacm:group nc:operation="create">  
                        <nacm:groupIdentity>nacm:guest</nacm:groupIdentity>  
                        <nacm:userName>fred</nacm:userName>  
                        <nacm:userName>barney</nacm:userName>  
                    </nacm:group>  
                </nacm:groups>  
            </nacm:nacm>  
        </nc:config>  
    </nc:edit-config>  
</nc:rpc>
```

### 8.7.4 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.

### 8.7.5 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 <allowedRights> 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 <allowedRights> 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.

### 8.7.6 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`.

### 8.7.7 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':

## Yuma Tools User Manual

- 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 operations is tagged as **nacm:secure** or **nacm:very-secure**, then deny access and exit.
    - b) If the <nacmNoRuleExecDefault> 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 <nacmNoRuleExecDefault> 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/rpcRule** entries that contain an <allowedGroup> 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 <allowedRights> leaf for the 'exec' bit.
    - If the 'exec' bit is found, then grant access, otherwise deny access, and exit.
    - If a matching <rpcRule> entry is not found, then proceed to step 8.
- Step 8) Check for any **/nacm/rules/moduleRule** entries that contain an <allowedGroup> 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 <allowedRights> leaf for the 'exec' bit.
    - If 'exec' bit is found, then grant access, otherwise deny access, and exit.
    - If a matching <moduleRule> entry is not found, then proceed to step 9.
- Step 9) If the <nacmNoRuleExecDefault> 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:

## Yuma Tools User Manual

- If the <nacmNoRuleReadDefault> 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/dataRule** entries that contain a <path> expression that evaluates to an XPath node-set that contains the requested database node, and the <allowedGroups> 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 <allowedRights> 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/moduleRule** entries that contain an <allowedGroup> 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 <allowedRights> 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 <nacmNoRuleReadDefault> 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 <nacmNoRuleWriteDefault> 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/dataRule** entries that contain a <path> expression that evaluates to an XPath node-set that contains the requested database node, and the <allowedGroups> 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 <allowedRights> 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/moduleRule** entries that contain an <allowedGroup> 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 <allowedRights> 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 <nacmNoRuleWriteDefault> leaf is set to 'permit' then grant access, otherwise deny access, and exit.

### 8.7.8 MODULE ACCESS CONTROL RULES

The **/nacm/rules/moduleRule** 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 <allowedAccess> 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 <moduleRule> 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.com/ns/nacm">
                <nacm:rules>
                    <nacm:moduleRule nc:operation="create">
                        <nacm:moduleName>nacm</nacm:moduleName>
                        <nacm:allowedRights>read write</nacm:allowedRights>
                        <nacm:allowedGroup>nacm:admin</nacm:allowedGroup>
                    </nacm:moduleRule>
                    <nacm:moduleRule nc:operation="create">
                        <nacm:moduleName>nacm</nacm:moduleName>
                        <nacm:allowedRights>read</nacm:allowedRights>
                        <nacm:allowedGroup>nacm:monitor</nacm:allowedGroup>
                    </nacm:moduleRule>
                </nacm:rules>
            </nacm:nacm>
        </nc:config>
    </nc:edit-config>
```

```
</nc:rpc>
```

## 8.7.9 RPC ACCESS CONTROL RULES

The **/nacm/rules/rpcRule** data structure is used to configure access for a specific RPC operation from a specific YANG module. If the module namespace URI for the **<rpcModuleName>** value is the same as the XML namespace used in the NETCONF PDU, and the **<rpcName>** 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 **<allowedAccess>** 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 **<rpcRule>** 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 **<allowedRights>** 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 **<moduleRule>** 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.com/ns/nacm">
                <nacm:rules>
                    <nacm:rpcRule nc:operation="create">
                        <nacm:rpcModuleName>netconf</nacm:rpcModuleName>
                        <nacm:rpcName>get-config</nacm:rpcName>
                        <nacm:allowedRights>exec</nacm:allowedRights>
                        <nacm:allowedGroup>nacm:admin</nacm:allowedGroup>
                        <nacm:allowedGroup>nacm:monitor</nacm:allowedGroup>
                    </nacm:rpcRule>
                    <nacm:rpcRule nc:operation="create">
                        <nacm:rpcModuleName>netconf</nacm:rpcModuleName>
                        <nacm:rpcName>get-config</nacm:rpcName>
                        <nacm:allowedRights />
                        <nacm:allowedGroup>nacm:guest</nacm:allowedGroup>
                    </nacm:rpcRule>
                </nacm:rules>
            </nacm:nacm>
        </nc:config>
    </nc:edit-config>
</nc:rpc>
```

```

</nacm:rpcRule>
</nacm:rules>
</nacm:nacm>
</nc:config>
</nc:edit-config>
</nc:rpc>

```

## 8.7.10 DATA ACCESS CONTROL RULES

The **/nacm/rules/dataRule** data structure is used to configure access for a specific set of database nodes (or notification payload nodes). If the requested node is contained within the node-set result of the <path> XPath 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 will be used.

If the 'exec' access bit is set in the <allowedRights> 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 <dataRule> 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.com/ns/nacm">
                <nacm:rules>
                    <nacm:dataRule nc:operation="create">
                        <nacm:name>if-1</nacm:name>

```

## Yuma Tools User Manual

```
<nacm:path
  xmlns:if="http://netconfcentral.com/ns/interfaces">
  /if:interfaces/if:interface
</nacm:path>
<nacm:allowedRights>read write</nacm:allowedRights>
<nacm:allowedGroup>nacm:admin</nacm:allowedGroup>
<nacm:comment>
  let admin group read and write all interfaces
</nacm:comment>
</nacm:dataRule>
<nacm:dataRule  nc:operation="create">
  <nacm:name>itf-2</nacm:name>
  <nacm:path
    xmlns:if="http://netconfcentral.com/ns/interfaces">
    /if:interfaces/if:interface[if:name='eth0']
</nacm:path>
  <nacm:allowedRights>read</nacm:allowedRights>
  <nacm:allowedGroup>nacm:monitor</nacm:allowedGroup>
  <nacm:comment>
    let monitor group read interface 'eth0'
  </nacm:comment>
</nacm:dataRule>
<nacm:dataRule  nc:operation="create">
  <nacm:name>itf-3</nacm:name>
  <nacm:path
    xmlns:if="http://netconfcentral.com/ns/interfaces">
    /if:interfaces
</nacm:path>
  <nacm:allowedRights/>
  <nacm:allowedGroup>nacm:guest</nacm:allowedGroup>
  <nacm:comment>
    do not let guest group read any interfaces info
  </nacm:comment>
</nacm:dataRule>
</nacm:rules>
</nacm:nacm>
</nc:config>
</nc:edit-config>
</nc:rpc>
```

## 8.8 Monitoring

## Yuma Tools User Manual

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**

<b>type</b>	<b>description</b>
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 <pre>&lt;filter type="subtree"&gt;     // some-xml-subtree &lt;/filter&gt;</pre> to retrieve portions of the <candidate> or <running> configurations.
XPath filtering	Use <pre>&lt;filter type="xpath"     select="expr"/&gt;</pre> to retrieve portions of the <candidate> or <running> configurations.

### 8.8.1 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.

## Yuma Tools User Manual

XML namespaces are optional to use:

- If there is no namespace in effect 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.com/ns/interfaces">
                <if:interface>
                    <if:name>eth0</if:name>
                </if:interface>
            </if:interfaces>
        </nc:filter>
    </nc:get>
</nc:rpc>
```

## Yuma Tools User Manual

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.com/ns/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.com/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>
```

## 8.8.2 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.com/ns/nacm">
      <nacm:rules>
        <nacm:dataRule>
          <nacm:name>nacm-tree</nacm:name>
```

## Yuma Tools User Manual

```
</nacm:dataRule>
<nacm:dataRule>
    <nacm:name>itf-1</nacm:name>
</nacm:dataRule>
</nacm:rules>
</nacm:nacm>
<t>xpath.1 xmlns:t="http://netconfcentral.com/ns/test">
    <t:name>barney</t:name>
</t>xpath.1>
<if:interfaces xmlns:if="http://netconfcentral.com/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:ietf-netconf-state xmlns:ns="urn:ietf:params:xml:ns:netconf:state">
    <ns:datasstores>
        <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: datasstores>
</ns:ietf-netconf-state>
<manageEvent:netconf xmlns:manageEvent="urn:ietf:params:xml:ns:netmod:notification">
```

## Yuma Tools User Manual

```
<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.com/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.com/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>
```

### 8.8.3 SYSTEM INFORMATION

### 8.8.4 INTERFACES INFORMATION

### 8.8.5 NETCONF STATE INFORMATION

## 8.9 Notifications

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

### 8.9.1 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 <stopTime> 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.

### 8.9.2 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 <startTlme> 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.

### 8.9.3 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.

### 8.9.4 <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 'notificationComplete' do not have this element. The namespace URI for this element is "http://netconfcentral.com/ns/system".

## 8.9.5 <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>
```

## 8.9.6 <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>
```

## 8.9.7 <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 <b>netconfd</b> 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>2009-07-29T17:31:22Z</ncEvent:eventTime>
    <sys:sysStartup xmlns:sys="http://netconfcentral.com/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.com/ns/system">1
</sys:sequence-id>
</ncEvent:notification>

```

## 8.9.8 <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.com/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.com/ns/system">4
</sys:sequence-id>
</ncEvent:notification>

```

## 8.9.9 <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)

## Yuma Tools User Manual

- 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.com/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.com/ns/system">5
  </sys:sequence-id>
</ncEvent:notification>
```

### 8.9.10 <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

## Yuma Tools User Manual

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.com/ns/system">
    <sys:userName>andy</sys:userName>
    <sys:sessionId>3</sys:sessionId>
    <sys:remoteHost>192.168.0.6</sys:remoteHost>
    <sys:edit>
      <sys:target>
```

## Yuma Tools User Manual

```
/nd:config/nacm:nacm:nacm:groups/nacm:group[nacm:groupIdentity=nacm:admin]
</sys:target>
<sys:operation>create</sys:operation>
</sys:edit>
</sys:sysConfigChange>
<sys:sequence-id
  xmlns:sys="http://netconfcentral.com/ns/system">7</sys:sequence-id>
</ncEvent:notification>
```

### 8.9.11 <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 identify 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 **ietf-netconf-state** 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.com/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&revision=2009-06-
      23&features=clock
    </sys:added-capability>
  </sys:sysCapabilityChange>
</ncEvent:notification>

```

```

</sys:sysCapabilityChange>
<sys:sequence-id
    xmlns:sys="http://netconfcentral.com/ns/system">8
</sys:sequence-id>
</ncEvent:notification>

```

## 8.9.12 <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

## Yuma Tools User Manual

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.com/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.com/ns/system">9
  </sys:sequence-id>
</ncEvent:notification>
```

## 9 CLI Reference

The Yuma programs uses command line interface (CLI) parameters to control program behavior.

Many parameters are supported by more than one program, such as **--log**.

The following sections document all the Yuma CLI parameters, in alphabetical order.

### 9.1 --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: <b>enforcing</b> <b>permissive</b> <b>disabled</b> <b>off</b>
Default:	enforcing
Min Allowed:	0
Max Allowed:	1
Supported by:	netconfd
Example:	<pre>netconfd \ --access-control=permissive</pre>

### 9.2 --autocomp

The **--autocomp** parameter controls automatic completion of command and parameter names.

## Yuma Tools User Manual

If true, then the program will attempt to find a partial match by comparing only the number of characters entered. For example '--log-l=debug' is the same as '--log-level=debug'.

If 'false', then command and parameter names must match exactly.

### --autocomp parameter

Syntax	boolean (true or false)
Default:	TRUE
Min Allowed:	0
Max Allowed:	1
Supported by:	yangcli
Example:	<code>yangcli --autocomp=false</code>

## 9.3 --autohistory

The **--autohistory** parameter controls automatic loading and saving of the command line history buffer in the **yangcli** program..

If true, then when the program starts, the default command line history buffer will be loaded, and the previous set of commands will be available with the **history** and **recall** commands.

If 'false', then the command line history buffer will not be loaded and saved automatically.

The default location for this file is `~/.yuma/.yangcli_history`.

### --autohistory parameter

Syntax	boolean (true or false)
Default:	TRUE
Min Allowed:	0
Max Allowed:	1
Supported by:	yangcli
Example:	<code>yangcli --autohistory=false</code>

## 9.4 --autoload

The **--autoload** parameter controls automatic loading of YANG modules, as needed.

If true, then the program will attempt to find a needed YANG module.

If 'false', then YANG modules must be loaded manually.

### --autoload parameter

Syntax	boolean (true or false)
Default:	TRUE
Min Allowed:	0
Max Allowed:	1
Supported by:	yangcli
Example:	<code>yangcli --autoload=false</code>

## 9.5 --bad-data

The **--bad-data** parameter controls how invalid parameter input is handled by the program.

- **ignore**: Use invalid parameter values. Use with caution.
- **warn**: Issue a warning message, but use the invalid data anyway.
- **check**: Prompt the user interactively, and check if the invalid data should be used, or a new value re-entered.
- **error**: Do not use invalid parameter values.

### --bad-data parameter

Syntax	enumeration: <b>ignore</b> <b>warn</b> <b>check</b> <b>error</b>
Default:	check
Min Allowed:	0
Max Allowed:	1
Supported by:	yangcli
Example:	<code>yangcli ----bad-data=warn</code>

## 9.6 --batch-mode

The **--batch-mode** parameter specifies that the interactive CLI will not be used.

Instead, if a script is provided with the 'run-script' parameter, or a command provided with the 'run-command' parameter, then it will be executed automatically before the program exits. This mode can be used to invoke NETCONF commands from Unix shell scripts.

### --batch-mode parameter

Syntax	empty
--------	-------

Default:	none
Min Allowed:	0
Max Allowed:	1
Supported by:	yangcli
Example:	<code>yangcli --batch-mode \ --run-script=~/run-tests</code>

## 9.7 --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/<program-name>.conf
Min Allowed:	0
Max Allowed:	1
Supported by:	netconfd yangcli yangdiff yangdump
Example:	<code>yangdump testmod \ --config=~/testconf.conf</code>

## 9.8 --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:	<b>\$YUMA_DATAPATH</b> environment variable
Min Allowed:	0

Max Allowed:	1
Supported by:	netconfd yangcli yangdiff yangdump
Example:	<code>netconfd \ --datapath="~/work2:~/data"</code>

## 9.9 --default-module

The **--default-module** parameter specifies the module to search for identifiers, after the **netconf** and **ncx** modules have been checked. This allows a specific module to match identifier names first, before all modules are searched at once. This can avoid a collision if the same identifier value is used in more than one module.

### --default-module parameter

Syntax	string: module name without any path or file extension included.
Default:	none
Min Allowed:	0
Max Allowed:	1
Supported by:	yangcli
Example:	<code>yangcli --default-module=testmod</code>

## 9.10 --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. This 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: <b>report-all</b> <b>trim</b> <b>explicit</b>
Default:	report-all
Min Allowed:	0
Max Allowed:	1
Supported by:	netconfd
Example:	<code>netconfd \ --default-style=trim</code>

## 9.11 --defnames

The **--defnames** parameter causes the program output file to be named with the default name for the format, based on the input module name and revision date. Refer to the section on generating WEB documentation for details on specific file formats for HTML output.

If the **--output** parameter is present and represents an existing directory, then the default filename will be created in that directory, instead of the current directory.

This parameter is ignored if the **--format** parameter is missing.

### --defnames parameter

Syntax	boolean
Default:	FALSE
Min Allowed:	0
Max Allowed:	1
Supported by:	yangdump
Example:	<code>yangdump \ --defnames=true \ --subtree=~/workdir/ \ --format=html</code>

## 9.12 --dependencies

The **--dependencies** parameter causes information to be reported for the symbols that this [sub]module imports from other modules.

The following information is reported for each dependency:

- module name
- revision date

Example report for module 'yangdump':

```
dependencies:
```

```
import ncx 2009-06-12
import ncx-app-common 2009-04-10
import ncxtypes 2008-07-20
```

### **--dependencies parameter**

Syntax	empty
Default:	none
Min Allowed:	0
Max Allowed:	1
Supported by:	yangdump
Example:	<pre>yangdump \ --dependencies \ --module=test4</pre>

## **9.13 --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:	<pre>yangcli \ --module=test1 \ --deviation=test1_deviations</pre>

## 9.14 --difftype

The **--difftype** parameter controls how differences are displayed in the **yangdiff** program..

The allowed values are 'terse', 'normal', and 'revision'.

The basic report format is:

```
[add/delete/modify] field-name [field-value]
```

The '**terse**' option will include the names of the top-level fields that are different. The actual differences for modification lines ('M') are not printed.

```
M typedef C
D container test-D1
D leaf test-D
M container test33
```

The '**normal**' option will include any changes for any nested fields or objects. A brief description of the changes made in a modification line ('M') are printed. This is the default reporting mode.

```
M typedef C
M type
  M range from 'min .. 41 | 45' to 'min .. 41'
D container test-D1
D leaf test-D
M container test33
  D presence 'not a top-level mand...'
M choice f
  M case f1
    M leaf f1
      A if-feature 'X'
```

The '**revision**' option will generate the differences report in YANG revision-stmt format. For example:

```
revision 2009-09-10 {
  description "
    - Changed typedef C
      - Changed type
        - Changed range from 'min .. 41 | 45' to 'min .. 41'
    - Removed container test-D1
    - Removed leaf test-D
    - Changed container test33
      - Removed presence 'not a top-level mand...'
      - Changed choice f
        - Changed case f1
          - Changed leaf f1
            - Added if-feature 'X'
```

```
" ;
}
```

**difftype parameter**

Syntax	enumeration: <b>terse</b> <b>normal</b> <b>revision</b>
Default:	normal
Min Allowed:	0
Max Allowed:	1
Supported by:	yangdiff
Example:	<code>yangdiff --difftype=revision \ --new=test3a\ --old=~test3</code>

## 9.15 --display-mode

The **--display-mode** parameter controls how data is displayed in the program.

The qualified names (identifiers, identityref, XPath) within a text configuration can be generated several ways. This is needed because sibling nodes in different XML namespaces can have the same name.

**--display-mode values**

enum value	description	example
plain	no prefix, just local-name	sysBootError
prefix	XML prefix and local-name	sys:sysBootError
module	module name and local name	system:sysBootError
xml	XML format	<sys:sysBootError>

When saving configuration files in non-volatile storage, then it is suggested that only 'module' or 'xml' modes be used, since these are the only deterministic modes.

The set of XML prefixes in use at any one time is not persistent, and cannot be relied upon, unless the namespace declarations are saved (xml mode) or the module names are saved (module mode).

**display-mode parameter**

Syntax	enumeration: <b>plain</b> <b>prefix</b> <b>module</b> <b>xml</b>
--------	--

Default:	plain
Min Allowed:	0
Max Allowed:	1
Supported by:	yangcli
Example:	<code>yangcli --display-mode=module</code>

## 9.16 --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:	<code>netconfd --eventlog-size=20000</code>

## 9.17 --exports

The **--exports** parameter causes information to be reported for the symbols that this [sub]module exports to other modules.

The exports for the entire module are printed, unless the specified input file is a YANG submodule. In that case, just the exports in the submodule are reported.

It includes the following information, generated in this order:

- [sub]module name
- version
- source filespec
- namespace (module only)
- prefix (module only)
- belongs-to (submodule only)

- `typedefs`
- `groupings`
- `objects`
- `RPC operations`
- `notifications`
- `extensions`
- `features`

#### **--exports parameter**

Syntax	<code>empty</code>
Default:	<code>none</code>
Min Allowed:	<code>0</code>
Max Allowed:	<code>1</code>
Supported by:	<code>yangdump</code>
Example:	<pre>yangdump \ --exports \ --module=test3</pre>

## 9.18 --format

The **--format** parameter controls the type of **yangdump** conversion desired, if any.

If this parameter is missing, then no translation will be done, but the module will be validated, and any requested reports will be generated.

The following values are supported:

- **yin**
  - Generate standard YIN format (XML instance document)
- **xsd**
  - XSD 1.0 translation .
  - Data model XSD can be integrated with the with the NETCONF protocol XSD in RFC 4741.
- **html**
  - XHTML 1.0 translation.
- **yang**
  - Canonical YANG translation .
- **copy**
  - Copy with a new name, in canonical module naming format.
- **sql**
  - TBD: Generate a module specific SQL template
  - This option is reserved for future use.

- **sqldb**
  - Generate module specific SQL data for the netconfcentral.sql template.
  - This option is only available in the SDK version of Yuma.
- **h**
  - Generate a module specific **netconfd** agent instrumentation H file.
  - This option is only available in the SDK version of Yuma.
- **c**
  - Generate a module specific **netconfd** agent instrumentation C file.
  - This option is only available in the SDK version of Yuma.

**--format parameter**

Syntax	enumeration (xsd, sql, sqldb, html, yang, copy, h, c))
Default:	none
Min Allowed:	0
Max Allowed:	1
Supported by:	yangdump
Example:	<pre>yangdump test1 test2 \ --format=xsd \ --defnames=true \ --output=~/workdir</pre>

**9.19 --fixorder**

The **--fixorder** parameter controls whether PDU parameters will be automatically sent in NETCONF PDUs in the correct order.

If 'true', the schema-defined, canonical order will be used.

If 'false', the specified order that parameters are entered will be used for the PDU order as well.

**--fixorder parameter**

Syntax	boolean (true or false)
Default:	TRUE
Min Allowed:	0
Max Allowed:	1
Supported by:	yangcli
Example:	<code>yangcli --fixorder=false</code>

## 9.20 --header

The **--header** parameter controls whether YANG header contents will be compared in the **yangdiff** program.

### --header parameter

Syntax	boolean (true or false)
Default:	TRUE
Min Allowed:	0
Max Allowed:	1
Supported by:	yangdiff
Example:	<pre>yangdiff --header=false \ --old=~/saved-modules \ -- new=~/work</pre>

## 9.21 --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:	<code>yangdump --help</code>

## 9.22 --help-mode

## Yuma Tools User Manual

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 <b>--brief</b> <b>--normal</b> <b>--full</b>
Default:	normal
Min Allowed:	0
Max Allowed:	1
Supported by:	netconfd yangcli yangdiff yangdump
Example:	<code>yangdump --help --help-mode=full</code>

- **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.

## 9.23 --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.

## Yuma Tools User Manual

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:	<code>netconfd --hello-timeout=300</code>

## **9.24 --html-div**

The **--html-div** parameter controls whether **yangdump** HTML translation will generate a single <div> element, or an entire HTML document.

If HTML translation is requested, then setting this parameter to 'true' will cause the output to be a single <div> element, instead of an entire HTML file. If it is not requested (**--format=html**), then this parameter is ignored.

This parameter allows the HTML translation to be easily integrated within more complex WEB pages, but the proper CSS definitions need to be present for the HTML to render properly. It is suggested only HTML experts use this parameter.

Refer to the following stylesheet file for more details:

<http://www.netconfcentral.com/static/css/style.css>

The default filename extension will be '.div' instead of '.html' if this parameter is present. The contents will be well-formed XHTML 1.0, but without any namespace declarations.

### **--html-div parameter**

Syntax	boolean
Default:	FALSE
Min Allowed:	0
Max Allowed:	1
Supported by:	yangdump
Example:	<code>yangdump \     --html-div=true  --format=html     --module=test</code>

## 9.25 --html-toc

The **--html-toc** parameter controls how **yangdump** HTML translation will generate a table of contents for the HTML document.

If HTML translation is requested, then this parameter will cause the output to contain a bullet list TOC, a drop-down menu TOC, or none.

This option has no effect unless the **--format=html** parameter is also present.

The following values are supported:

- **none**
  - No TOC is generated.
- **plain**
  - A plain list-based TOC is generated
- **menu**
  - A suckerfish (Javascript) drop-down menu will be generated.
  - This is the default option.

Note that if the 'menu' enumeration is used, then the following javascript file must be available to the WEB server which is hosting the output HTML file:

```
http://www.netconfcentral.org/static/javascript/suckerfish.js
```

### --html-toc parameter

Syntax	enumeration (none, plain, menu)
Default:	menu
Min Allowed:	0
Max Allowed:	1
Supported by:	yangdump
Example:	<pre><code>yangdump \ --html-toc=plain --format=html --module=test</code></pre>

## 9.26 --identifiers

The **--identifiers** parameter causes information to be object identifier strings to be reported for the object, RPC operation, and notification definitions within the input module(s).

The following information is reported for each identifier:

- object type (e.g., leaf, container, rpc)
- absolute XPath expression for the object

Example report for module 'mysession':

```

identifiers:
  rpc /get-my-session
  container /get-my-session/output
  leaf /get-my-session/output/indent
  leaf /get-my-session/output/linesize
  leaf /get-my-session/output/with-defaults
  container /get-my-session/input
  rpc /set-my-session
  container /set-my-session/input
  leaf /set-my-session/input/indent
  leaf /set-my-session/input/linesize
  leaf /set-my-session/input/with-defaults
  container /set-my-session/output

```

### **--identifiers parameter**

Syntax	empty
Default:	none
Min Allowed:	0
Max Allowed:	1
Supported by:	yangdump
Example:	<pre>yangdump \ --identifiers \ --module=mysession</pre>

## 9.27 --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

**9.28 --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:	yangcli --indent=4

**9.29 --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&

## 9.30 --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&

## 9.31 --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: <b>off</b> <b>error</b> <b>warn</b> <b>info</b> <b>debug</b> <b>debug2</b> <b>debug3</b>
Default:	--info (--debug for DEBUG builds)
Min Allowed:	0
Max Allowed:	1
Supported by:	netconfd yangcli yangdiff yangdump
Example:	<code>netconfd --log-level=debug \ --log=~/server.log&amp;</code>

## 9.32 --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:	<code>netconfd --max-burst=100</code>

## 9.33 --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:	<b>\$YUMA_MODPATH</b> environment variable
Min Allowed:	0
Max Allowed:	1
Supported by:	netconfd yangcli yangdiff yangdump
Example:	<pre>yangdump \ --modpath="~/testmodules:~/modules: ~/trunk/netconf/modules" \ --module=~/test42.yang</pre>

**9.34 --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:	<pre>yangcli \ --module=test1 \ --module=test2</pre>

**9.35 --modversion**

The **--modversion** parameter causes the module name and revision date to be displayed for each module specified in the input.

Example output for module 'test':

```
modversion:  
module test 2009-06-10
```

### **--modversion parameter**

Syntax	empty
Default:	none
Min Allowed:	0
Max Allowed:	1
Supported by:	yangdump
Example:	<pre>yangdump \ --modversion \ --module=test</pre>

## 9.36 --new

The **--new** parameter specifies the YANG file or directory containing the new revision to be compared in the **yangdiff** program.

If this parameter indicates a filename, then it represents the YANG source module name to compare as the newer of the two revisions.

If this parameter indicates a directory (and the 'old' parameter indicates a filename), then it will be used to search for a file with the same name as the 'new' parameter.

If the 'old' parameter identifies a directory as well (and the 'no-subdirs' parameter is present), then the modules within the 'new' directory will be compared to files with the same name in the 'old' directory.

If the **--subdirs** parameter is "true", then all sub-directories within the 'src' directory will also be checked.

If this string begins with a '~' character, then a username is expected to follow or a directory separator character. If it begins with a '\$' character, then an environment variable name is expected to follow.

```
~/some/path ==> <my-home-dir>/some/path
~fred/some/path ==> <fred-home-dir>/some/path
$workdir/some/path ==> <workdir-env-var>/some/path
```

This parameter must be present unless the **--help** or **--version** parameters are present.

### **--new parameter**

Syntax	string (module or directory specification. length 1 .. 4095)
Default:	none
Min Allowed:	1

Max Allowed:	1
Supported by:	yangdiff
Example:	<pre>yangdiff \ --new=test3a --difftype=terse --old=test3\</pre>

## 9.37 --objview

The **--objview** parameter specifies how objects will be generated during translation, for HTML and canonical YANG file translations.

The enumeration values are:

- **raw**
  - output includes augment and uses clauses, not the expanded results of those clauses.
- **cooked**
  - output does not include augment or uses clauses, just the objects generated from those clauses.

The default mode is the 'raw' view.

XSD output is always 'cooked', since refined groupings and locally-scoped definitions are not supported in XSD. This parameter is not implemented for other values of the **--format** parameter.

### --objview parameter

Syntax	enumeration (raw, cooked)
Default:	raw
Min Allowed:	0
Max Allowed:	1
Supported by:	yangdump
Example:	<pre>yangdump \ --objview=cooked \ --module=test \ --format=html</pre>

## 9.38 --old

The **--old** parameter specifies the YANG file or directory containing the older revision to be compared in the **yangdiff** program.

If this parameter indicates a filename, then it represents the YANG source module name to compare as the older of the two revisions.

If this parameter indicates a directory (and the 'old' parameter indicates a filename), then it will be used to search for a file with the same name as the 'new' parameter.

## Yuma Tools User Manual

If this string begins with a '~' character, then a username is expected to follow or a directory separator character. If it begins with a '\$' character, then an environment variable name is expected to follow.

```
~/some/path ==> <my-home-dir>/some/path  
~fred/some/path ==> <fred-home-dir>/some/path  
$workdir/some/path ==> <workdir-env-var>/some/path
```

This parameter must be present unless the **--help** or **--version** parameters are present.

### **--old parameter**

Syntax	string (module or directory specification. length 1 .. 4095)
Default:	none
Min Allowed:	1
Max Allowed:	1
Supported by:	yangdiff
Example:	<pre>yangdiff \   --old=test3   --difftype=terse   --new=test3a\   \</pre>

## 9.39 --output

The **--output** parameter specifies where the output files generated by the program will be stored.

- The default is STDOUT if this parameter is not specified and the **--defnames** parameter is set to 'false'.
- If this parameter represents an existing directory, then the **--defnames** parameter will be set to 'true' by default.
- If this parameter represents a file name, then the **--defnames** parameter will be ignored, and all translation output will be directed to the specified file.
- If this string begins with a '~' character, then a username is expected to follow or a directory separator character. If it begins with a '\$' character, then an environment variable name is expected to follow.

```
~/some/path ==> <my-home-dir>/some/path
```

```
~fred/some/path ==> <fred-home-dir>/some/path
```

```
$workdir/some/path ==> <workdir-env-var>/some/path
```

- If the target specified in this parameter **does not** exist:

## Yuma Tools User Manual

- If there is only one file to be output, then this parameter is used as the file name.
- If there are multiple files to be output, then this parameter is used as a directory name. A new directory will be created, if it is needed.
- If the target specified in this parameter **does** exist:
  - If there is only one file to be output:
    - If the existing target is also a file, then the current file is over-written.
    - If the existing target is a directory, then the output file will be created in this directory.
  - If there are multiple files to be output:
    - If the existing target is a file, then an error will be generated instead of the output files.
    - If the existing target is a directory, then the output files will be created in the specified directory.
- Use a trailing path separator character to force this parameter value to be treated as a path specification (e.g., '~/workfiles/').
- This parameter is ignored if the **--format** parameter is missing.

### **--output parameter**

Syntax	string (path or file specification)
Default:	none
Min Allowed:	0
Max Allowed:	1
Supported by:	yangdump yangdiff
Example:	<pre>yangdump \   --output=~/html-files   --subtree=testfiles \   --format=html</pre>

## **9.40 --password**

The **--password** parameter specifies the password string that should be used when a NETCONF session is connected during the startup of the program.

### **--password parameter**

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

Example:	<code>yangcli --server=myserver \ --password=yangrocks \ --user=andy</code>
----------	---

## 9.41 --port

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

In the **yangcli** program, this parameter specifies the TCP port number to use when a NETCONF session is created during the startup of the program. The parameter can only be entered once.

In the **netconfd** program, this parameter specifies a TCP port number 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:	1 (4 in netconfd)
Supported by:	netconfd yangcli
Example:	<code>netconfd --port=22 \ --port=830</code>

## 9.42 --runpath

The **--runpath** parameter specifies the directory search path to use while searching for script 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\_RUNPATH** environment variable, if it is present.

### --runpath parameter

Syntax	string: list of directory specifications
Default:	<b>\$YUMA_RUNPATH</b> environment variable
Min Allowed:	0
Max Allowed:	1
Supported by:	netconfd yangcli yangdiff yangdump

Example:	<code>yangcli \ --runpath="~/testscripts"</code>
----------	--

## 9.43 --run-command

The **--run-command** parameter specifies a command will be invoked upon startup.

In the **yangcli** program, if the auto-connect parameters are provided, then a session will be established before running the command.

### --run-command parameter

Syntax	string
Default:	none
Min Allowed:	0
Max Allowed:	1
Supported by:	yangcli
Example:	<code>yangcli \ --run-command="history load=~/test3-history"</code>

## 9.44 --run-script

The **--run-script** parameter specifies a script will be invoked upon startup.

In the **yangcli** program, if the auto-connect parameters are provided, then a session will be established before running the script.

### --run-script parameter

Syntax	string (file specification)
Default:	none
Min Allowed:	0
Max Allowed:	1
Supported by:	yangcli
Example:	<code>yangcli \ --run-script="test3-start"</code>

## 9.45 --server

The **--server** parameter specifies the IP address or DNS name of the NETCONF server that should be connected to automatically, when the program starts.

### **--server parameter**

Syntax	string:IP address or DNS name
Default:	none
Min Allowed:	0
Max Allowed:	1
Supported by:	yangcli
Example:	<code>yangcli --server=myserver</code>

## 9.46 --show-errors

The **--show-errors** parameter causes the yangdump program to list all the error and warning numbers, and the default message for each one.

The 3 digit number, followed by the message string, will be printed 1 per line.

After this is done, the **yangdump** program will exit.

This parameter can be combined with the **--help** parameter. The **--version** parameter has no affect if this parameter is present. The program version string will be printed in both cases.

The NETCONF error-tag values are used directly when no other error number is appropriate. These error numbers are as follows:

```

257  resource in use
258  invalid value
259  too big
260  missing attribute
261  bad attribute
262  unknown attribute
263  missing element
264  bad element
265  unknown element
266  unknown namespace
267  access denied
268  lock denied
269  resource denied
270  rollback failed
271  data exists
272  data missing
273  operation not supported
274  operation failed

```

**-show-errors parameter**

Syntax	empty
Default:	none
Min Allowed:	0
Max Allowed:	1
Supported by:	yangdump
Example:	<code>yangdump --show-errors</code>

**9.47 --simurls**

The **--simurls** parameter specifies how URL strings are generated for HTML links.

If HTML translation is requested, then setting this parameter to 'true' will cause the format of URLs within links to be generated in simplified form, for WEB development engines such as CherryPy, which support this format.

- Normal URL format (false):
  - `http://example.com/mymodule.html?parm1=foo&parm2=bar#frag`
- Simplified URL format (true):
  - **`http://example.com/mymodule/foo/bar#frag`**

**--simurls parameter**

Syntax	boolean
Default:	FALSE
Min Allowed:	0
Max Allowed:	1
Supported by:	yangdump
Example:	<code>yangdump \     --simurls=true \     --format=html \     --module=test</code>

**9.48 --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: <b>--no-startup</b> <b>--startup=filespec</b>
Default:	first startup-cfg.xml in the data path
Min Allowed:	0
Max Allowed:	1
Supported by:	netconfd
Example:	<code>netconfd \     --startup=\$TESTDIR/testrun</code>

**9.49 --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:	<code>yangdump \     --subdirs=false \     --subtree=/testpath</code>

## 9.50 --subtree

The **--subtree** parameter is a leaf-list of path specifications that may contain YANG files.

It must be a string value that represents a path specification of the directory subtree to use.

All of the YANG source modules contained in the specified directory sub-tree will be processed.

Note that symbolic links are not followed during the directory traversal. Only real directories will be searched and regular files will be checked as modules. Processing will continue to the next file if a module contains errors.

If this string begins with a '~' character, then a username is expected to follow or a directory separator character. If it begins with a '\$' character, then an environment variable name is expected to follow.

If the **--subdirs=false** parameter is present, then only the top-level directory specified by this parameter will be searched for YANG files.

If the **--subdirs=true** parameter is present, then all the directories contained within the one specified by this parameter will also be searched for YANG files. The exceptions are:

- any directory beginning with a dot character ('.'), such as '.svn'
- any directory named 'CVS'

Examples:

```
~/some/path ==> <my-home-dir>/some/path
~fred/some/path ==> <fred-home-dir>/some/path
$workdir/some/path ==> <workdir-env-var>/some/path
```

### subtree parameter

Syntax	string: path specification
Default:	none
Min Allowed:	0
Max Allowed:	unlimited
Supported by:	yangdiff yangdump
Example:	<pre>yangdump \ --format=html \ --subtree=~/testmods --subtree=./workdir --output=./yang-html-files</pre>

## 9.51 --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	<i>string</i>
Default:	superuser
Min Allowed:	0
Max Allowed:	1
Supported by:	netconfd
Example:	<code>netconfd --superuser=admin</code>

## **9.52 --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: <b>running</b> <b>candidate</b>
Default:	candidate
Min Allowed:	0
Max Allowed:	1
Supported by:	netconfd
Example:	<code>netconfd --target=running</code>

## **9.53 --timeout**

The **--timeout** parameter specifies the number of seconds that should pass before giving up on a response during a NETCONF session. The value zero means wait forever (no timeout).

**--timeout parameter**

Syntax	uint32 (0 for no timeout; otherwise number of seconds to wait)
Default:	30 seconds
Min Allowed:	0
Max Allowed:	1
Supported by:	yangcli
Example:	<code>yangcli --timeout=0</code>

## 9.54 --unified

The **--unified** parameter indicates if submodules should be combined into the main module for **yangdump** translation output. Any include statements will be replaced by the module definitions contained within the submodule.

If set to 'true', then submodules will be processed within the main module, in a unified report, instead of separately, one report for each file.

For translation purposes, this parameter will cause any sub-modules to be treated as if they were defined in the main module. Actual definitions will be generated instead of an 'include' statement, for each submodule.

If this mode is selected, then submodules will be ignored:

- If entered with the **--module** parameter explicitly.
- If found when searching for main YANG files to process in a directory, e.g., for the **--subtree** parameter.

If set to 'false', then a separate output file is generated for each input file, so that XSD output and other reports for a main module will not include information for submodules.

**--unified parameter**

Syntax	boolean
Default:	FALSE
Min Allowed:	0
Max Allowed:	1
Supported by:	yangdump
Example:	<pre>yangdump \ --unified=true \ --format=html \ --subtree=\$PROJECT_X/modules \ --defnames=true \ --output=\$PROJECT_X/html</pre>

## 9.55 --urlstart

The **--urlstart** parameter specifies the string to start all URLs during **yangdump** HTML translation.

If present, then this string will be used to start all HREF links and URLs generated for SQL and HTML translation. It is expected to be a URL ending with a directory path. The trailing separator '/' will be added if it is missing.

If not present (the default), then relative URLs, starting with the file name will be generated instead.

For example, if this parameter is set to the following string:

- `http://acme.com/public`

The URL generated for the 'bar' type on line 53, in the module FOO (version 2008-01-01) would be:

- if **--versionnames=false**:
  - `http://acme.com/public/FOO.html#bar.53`
- if **--versionnames=true** (default):
  - `http://acme.com/public/FOO_2008-01-01.html#bar.53`

### --urlstart parameter

Syntax	string (URL format)
Default:	none
Min Allowed:	0
Max Allowed:	1
Supported by:	yangdump
Example:	<pre>yangdump \ --urlstart="/example.com/public/" \ --unified=true \ --format=html \ --subtree=\$PROJECT_X/modules \ --defnames=true \ --output=\$PROJECT_X/html</pre>

## 9.56 --user

The **--user** parameter specifies the user name string on the NETCONF server that should be used when establishing a NETCONF session.

### --user parameter

Syntax	string:user name
Default:	none
Min Allowed:	0
Max Allowed:	1

Supported by:	yangcli
Example:	<code>yangcli --user=admin \ --server=myserver</code>

## 9.57 --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:	<code>netconfd --usexmlorder</code>

## 9.58 --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:

```
yangdump 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:	yangdump --version

## 9.59 --versionnames

The **--versionnames** parameter indicates that revision dates should not be used when constructing any output YANG module file names.

If present, the default filenames will not contain the module version string. Normally, the [sub]module name and version string are both used to generate a default file name, when the **--defnames** output parameter is set to 'true'. This parameter will cause filenames to be generated which do not contain the revision date string.

### --versionnames parameter

Syntax	boolean
Default:	TRUE
Min Allowed:	0
Max Allowed:	1
Supported by:	yangdump
Example:	<pre>yangdump \     --versionnames=false \     --subtree=/testpath \     --defnames=true \     --output=~/work3 \     --format=html</pre>

## 9.60 --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 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:	yangcli --warn-idlen=50

## 9.61 --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 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:	yangcli --warn-linelen=79

## 9.62 --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:	<code>netconfd --warn-off=435 # revision order not descending</code>
----------	--

## 9.63 --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:	<code>netconfd --with-startup=true</code>

## 9.64 --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:	<code>netconfd --with-validate=false</code>

## 9.65 --xsd-schemaloc

The **--xsd-schemaloc** parameter specifies that **yangdump** XSD translations should include the 'schemaLocation' attribute.

## Yuma Tools User Manual

If present, then the 'schemaLocation' attribute will be generated during XSD translation. This will be done for the module being processed, and any modules that are imported into that module.

If not present (the default), then the 'schemaLocation' attribute is not generated during XSD translation.

Relative URLs for include and import directives will be generated, starting with the file name.

For example, if this parameter is set to the following string:

```
http://example.com/public'
```

The 'schemaLocation' XSD for the module test3 (version 10-19-2008) would be:

- If **--versionnames=false**:
  - xsi:schemaLocation='http://netconfcentral.com/ns/test3 http://example.com/public/test3.xsd'
- if **--versionnames=true** (default):
  - xsi:schemaLocation='http://netconfcentral.com/ns/test3 http://example.com/public/test3\_2008-10-19.xsd'

### **--xsd-schemaloc parameter**

Syntax	string (URL formal)
Default:	none
Min Allowed:	0
Max Allowed:	1
Supported by:	yangdump
Example:	<pre>yangdump \ --xsd-schemaloc="http://example.com" --format=xsd --module=test3 --defnames=true</pre>

## 9.66 --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:	<b>\$YUMA_HOME</b> environment variable
Min Allowed:	0
Max Allowed:	1
Supported by:	netconfd yangcli yangdiff yangdump
Example:	<pre>netconfd \ --yuma-home=~/sw/netconf \ --log=~/server.log&amp;</pre>

## 10 Error Reference

All Yuma programs use the same set of error numbers and error messages.

Error numbers are 3 digit unsigned integers in the range 1 to 999. The number 0 is reserved for the NO\_ERR constant, which is the same as the <ok/> status returned by the server.

#### **Error Number Types**

range	description
100 to 199	system errors
200 to 399	user errors
400 to 899	warnings
900 to 999	informational messages

### 10.1 Error Messages

The current list of error numbers and default error messages can be obtained with the yangdump program **--show-errors** parameter.

The default error message can be replaced for some error conditions with the YANG error-message statement.

The following list shows the default error messages for all error numbers currently in use.

0	ok
1	EOF reached

## Yuma Tools User Manual

```
2    NULL pointer
3    malloc failed
4    invalid internal value
5    internal buffering failed
6    invalid queue deletion
7    wrong init sequence
8    queue node not header
9    queue node not data
10   invalid queue header
11   entry already queued
12   too many entries
13   libxml2 operation failed
100  cannot open file
101  cannot read file
102  cannot close file
103  cannot write file
104  cannot change directory
105  cannot stat file
106  buffer overflow error
107  cannot delete file
108  cannot access file
109  db connect failed
110  db entry exists
111  db not found
112  db query failed
113  db delete failed
114  wrong checksum
115  wrong tag type
116  db read failed
117  db write failed
118  db init failed
119  beep init failed
120  beep init nc failed
121  xml reader internal
122  open directory failed
123  read directory failed
200  no config file
201  no source file
202  POST read input
203  bad drive
204  bad path
205  bad filename
206  duplicate value pair
```

## Yuma Tools User Manual

```
207 page not handled
208 page access denied
209 missing form params
210 invalid form state
211 duplicate namespace
212 xml reader start failed
213 xml reader read failed
214 wrong XML node type
215 xml reader null name
216 xml reader null value
217 xml reader wrong name
218 xml reader wrong value
219 xml reader wrong element
220 xml reader extra nodes
221 xml reader EOF
222 wrong length
223 entry exists
224 duplicate entry
225 not found
226 missing file
227 unknown parameter
228 invalid name
229 unknown namespace
230 wrong namespace
231 wrong data type
232 wrong value
233 missing parameter
234 extra parameter
235 empty value
236 module not found
237 max length exceeded
238 invalid token
239 unended quoted string
240 read failed
241 invalid number
242 invalid hex number
243 invalid real number
244 EOF reached
245 wrong token type
246 wrong token value
247 buffer overflow
248 invalid range
249 overlapping range
```

## Yuma Tools User Manual

```
250    definition not found
251    definition segment not found
252    type not allowed in index
253    index type not found
254    type not mdata
255    meta-data not allowed
256    top not found
257    resource in use
258    invalid value
259    too big
260    missing attribute
261    bad attribute
262    unknown attribute
263    missing element
264    bad element
265    unknown element
266    unknown namespace
267    access denied
268    lock denied
269    resource denied
270    rollback failed
271    data exists
272    data missing
273    operation not supported
274    operation failed
275    partial operation
276    wrong namespace
277    wrong node depth
278    wrong owner
279    wrong element
280    wrong order
281    extra node
282    wrong node type
283    expecting complex node type
284    expecting string node type
285    wrong data type
286    wrong data value
287    invalid number length
288    value not in range
289    wrong number type
290    invalid enum value
291    value not in set
292    extra list string found
```

## Yuma Tools User Manual

```
293    unknown object
294    extra parameter instance
295    extra case in choice
296    missing mandatory choice
297    wrong config state
298    unknown application
299    unknown data type
300    access control violation
301    config locked
302    wrong config state
303    max-access exceeded
304    wrong index type
305    wrong instance type
306    missing index component
307    config not found
308    extra attribute instance(s) found
309    required attribute not found
310    required value instance not found
311    extra value instance(s) found
312    target is read only
313    invalid pattern
314    wrong version
315    connect failed
316    unknown host
317    session failed
318    authentication failed
319    end of comment not found
320    invalid string concatenation
321    import not found
322    missing typedef sub-section
323    restriction not allowed for this type
324    specified refinement not allowed
325    definition loop detected
326    default case contains mandatory object(s)
327    import loop
328    include loop
329    expecting module
330    expecting submodule
331    undefined prefix
332    imported module has errors
333    pattern match failed
334    invalid data type change
335    mandatory object not allowed
```

## Yuma Tools User Manual

```
336 unique-stmt test failed
337 max-elements exceeded
338 min-elements not reached
339 must-stmt test failed
340 data restriction violation
341 missing instance for insert operation
342 object not config
343 invalid conditional object
344 using obsolete definition
345 invalid augment target
346 duplicate refine sub-clause
347 invalid deviate sub-clause
348 invalid XPath expression syntax
349 invalid instance-identifier syntax
350 require-instance test failed
351 key or select attribute not allowed
352 invalid unique-stmt node
353 invalid duplicate import-stmt
354 invalid duplicate include-stmt
355 ambiguous command
356 unknown module
357 unknown version
358 value not supported
359 leafref path loop
360 variable not found
361 variable is read-only
362 decimal64 base number overflow
363 decimal64 fraction precision overflow
364 when-stmt tested false
365 no matches found
366 missing refine target
367 candidate cannot be locked, discard-changes needed
368 timeout occurred
400 duplicate source
401 include file not found
402 invalid command line value
403 invalid command line option
404 command line option unknown
405 invalid command line syntax
406 missing command line value
407 invalid form input
408 invalid form
409 no instance found
```

## Yuma Tools User Manual

```
410 session closed by remote peer
411 duplicate import
412 duplicate import with different prefix value
413 local typedef not used
414 local grouping not used
415 import not used
416 duplicate unique-stmt argument
417 statement ignored
418 duplicate include
419 include not used
420 revision date before 1970
421 revision date in the future
422 enum value order
423 bit position order
424 invalid status for child node
425 duplicate sibling node name from external augment
426 duplicate if-feature statement
427 using deprecated definition
428 XPath object predicate check limit reached
429 empty XPath result in must or when expr
430 no ancestor node available
431 no parent node available
432 no child node available
433 no descendant node available
434 no nodes available
435 bad revision-stmt order
436 duplicate prefix
437 identifier length exceeded
438 display line length exceeded
439 received unknown capability
440 invalid module capability URI
441 unknown child node, using anyxml
442 invalid value used for parm
443 changing object type to string
444 using a reserved name
445 conf file parm already exists
446 no valid revision statements found
447 dependency file has errors
448 top-level object is mandatory
```