

Yuma yangdump Manual

YANG-Based Unified Modular Automation Tools

YANG Module Compiler

Version 2.2

Last Updated: January 26, 2012

Table Of Contents

Yuma yangdump Manual

1	Preface.....	3
1.1	Legal Statements.....	3
1.2	Additional Resources.....	3
1.2.1	WEB Sites.....	3
1.2.2	Mailing Lists.....	4
1.3	Conventions Used in this Document.....	4
2	yangdump.....	5
2.1	Introduction.....	5
2.1.1	Features.....	6
2.1.2	Starting yangdump.....	8
2.1.3	Stopping yangdump.....	9
2.1.4	Configuration Parameter List.....	9
2.2	Validating YANG Files.....	11
2.2.1	Yangdump Validation Messages.....	12
2.2.2	Validation Example.....	12
2.3	Translating YANG to Other Formats.....	13
2.3.1	YIN Format.....	14
2.3.2	HTML Translation.....	15
2.3.3	XSD Translation.....	18
2.3.4	SQL Translation.....	21
2.3.5	SQL Documentation Database Translation.....	21
2.3.6	Canonical YANG Translation.....	21
2.3.7	Copy and Rename YANG Files.....	22
2.3.8	Agent Instrumentation H File Generation.....	22
2.3.9	Agent Instrumentation C File Generation.....	22
3	CLI Reference.....	24
3.1	--config.....	24
3.2	--defnames.....	24
3.3	--dependencies.....	25
3.4	--deviation.....	25
3.5	--exports.....	26
3.6	--feature-code-default.....	27
3.7	--feature-disable.....	27
3.8	--feature-dynamic.....	28
3.9	--feature-enable.....	28
3.10	--feature-enable-default.....	29
3.11	--feature-static.....	29
3.12	--format.....	30
3.13	--help.....	31
3.14	--help-mode.....	32
3.15	--html-div.....	33
3.16	--html-toc.....	33
3.17	--identifiers.....	34
3.18	--indent.....	35

Yuma yangdump Manual

3.19	--log.....	35
3.20	--log-append.....	36
3.21	--log-level.....	36
3.22	--modpath.....	37
3.23	--module.....	37
3.24	--modversion.....	38
3.25	--objview.....	38
3.26	--output.....	39
3.27	--show-errors.....	40
3.28	--simurls.....	41
3.29	--stats.....	41
3.30	--subdirs.....	44
3.31	--subtree.....	44
3.32	--totals.....	45
3.33	--tree-identifiers.....	46
3.34	--unified.....	46
3.35	--urlstart.....	47
3.36	--version.....	48
3.37	--versionnames.....	49
3.38	--warn-idlen.....	49
3.39	--warn-linelen.....	49
3.40	--warn-off.....	50
3.41	--xsd-schemaloc.....	50
3.42	--yuma-home.....	51

1 Preface

1.1 Legal Statements

Copyright 2009 - 2012, Andy Bierman, All Rights Reserved.

1.2 Additional Resources

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

Yuma Installation Guide

Other documentation includes:

Yuma Quickstart Guide

Yuma User Manual

Yuma netconfd Manual

Yuma yangcli Manual

Yuma yangdiff Manual

Yuma Developer Manual

To obtain additional support you may join the yuma-users group on sourceforge.net and send email to this e-mail address:

yuma-users@lists.sourceforge.net

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

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

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

The following section lists the resources available at this time.

1.2.1 WEB SITES

- **Netconf Central**
 - <http://www.netconfcentral.org/>
 - Yuma Home Page
 - Free information on NETCONF and YANG, tutorials, on-line YANG module validation and documentation database
- **Yuma SourceFource OpenSource Project**
 - <http://sourceforge.net/projects/yuma/>

Yuma yangdump Manual

- Download Yuma source and binaries; project forums and help
- **Yang Central**
 - <http://www.yang-central.org>
 - Free information and tutorials on YANG, free YANG tools for download
- **NETCONF Working Group Wiki Page**
 - <http://trac.tools.ietf.org/wg/netconf/trac/wiki>
 - Free information on NETCONF standardization activities and NETCONF implementations
- **NETCONF WG Status Page**
 - <http://tools.ietf.org/wg/netconf/>
 - IETF Internet draft status for NETCONF documents
- **libsmi Home Page**
 - <http://www.ibr.cs.tu-bs.de/projects/libsmi/>
 - Free tools such as smidump, to convert SMIV2 to YANG

1.2.2 MAILING LISTS

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

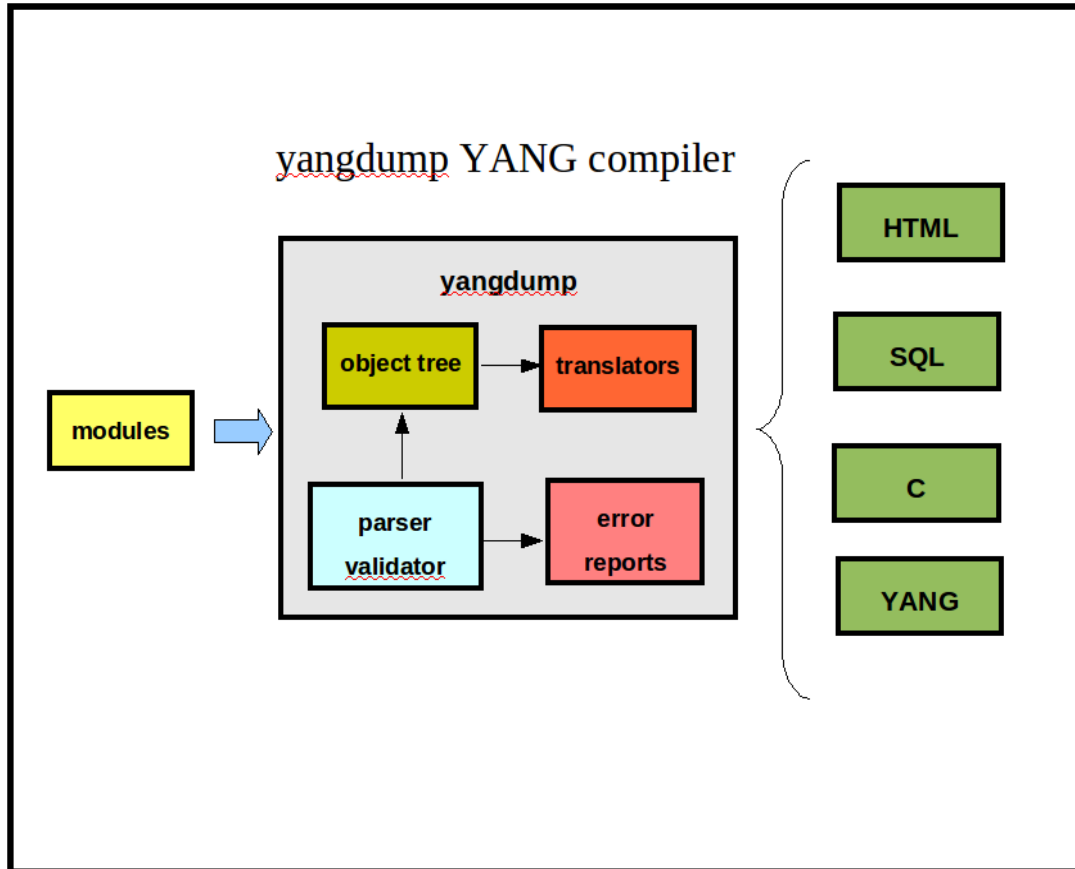
1.3 Conventions Used in this Document

The following formatting conventions are used throughout this document:

Documentation Conventions

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

2 yangdump



2.1 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.

The `yangdumpcode` program is the same as `yangdump`, except that the `--format` parameter allows more options to generate code for Yuma applications.

The following `--format` parameter values are allowing `yangdumpcode`, but not `yangdump`:

- **format=c (Experimental)**
 - C file for server instrumentation library for a YANG module
- **format=h (Experimental)**
 - H file for server instrumentation library for a YANG module
- **format=sqldb**
 - SQL database contents for the Yuma WEB docs server for a YANG module

Yuma yangdump Manual

- **format=tg2 (TBD - Not Supported)**
 - Turbogears 2 source code files for Yuma WEB applications for a YANG module

2.1.1 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.
- 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:

Yuma yangdump Manual

- 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.
- 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:

Yuma yangdump Manual

- 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).
 - 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.
- Generates C source code stubs for **netconfd** server instrumentation libraries, for dynamic loading of a YANG module.
- Generates informational reports on the contents of a YANG file:
 - Imported modules (dependencies).
 - Exported symbols (exports).
 - Object name tree (identifiers or tree-identifiers)
 - Module revision date (modversion)
 - YANG module metrics reports (stats and totals)

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

- naming conflicts report
- preserve and translate YANG comments
- preserve or reformat YANG string concatenation for **-objview=cooked**
 - YANG and HTML output will preserve original string token sequences for **-objview=raw** (the default). Most text string fields are preserved if **-format=html** or **--format=yang**

2.1.2 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.

Yuma yangdump Manual

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
```

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
```

2.1.3 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.

2.1.4 CONFIGURATION PARAMETER LIST

Yuma yangdump Manual

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.
--datapath	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.
--deviation	Species one or more YANG modules to load as deviations.
--exports	Generate the module exports report.
--feature-code-default	Specifies if a feature should use static or dynamic code by default
--feature-disable	Leaf list of features to disable
--feature-dynamic	Specifies a feature that uses dynamic code
--feature-enable	Specifies a feature that should be enabled
--feature-enable-default	Specifies if a feature should be enabled or disabled by default
--feature-static	Specifies a feature that uses static code
--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 gnerated.
--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.

Yuma yangdump Manual

--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.
--stats	Control YANG usage statistics reporting for each input module.
--subdirs	Controls whether sub-directories are searched for YANG files.
--subtree	Specifies one or more directory subtrees to search for YANG modules.
--tree-identifiers	Generate the module object identifier local-names report in tree format.
--totals	Controls statistics summary reporting for a set of input 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 \$YUMA_HOME project root to use when searching for files.

2.2 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.

Yuma yangdump Manual

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.

2.2.1 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.
- 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

2.2.2 VALIDATION EXAMPLE

Yuma yangdump Manual

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
```

2.3 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.

2.3.1 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"
  xmlns="urn:ietf:params:xml:ns:yang:yin:1"
  xmlns:t4="http://netconfcentral.org/ns/test4">
  <namespace uri="http://netconfcentral.org/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>
```

```

    </leaf>
  </list>
</module>

```

2.3.2 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>
    <li><a href="#">Objects</a>
      <ul>
        <li class="daddy"><a href="#a.15">a</a>
          <ul>

```

Yuma yangdump Manual

```
        <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.org/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>;
  }

  <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>
}
```

Yuma yangdump Manual

```
    <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

```

2.3.3 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 yangdump 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.org/ns/test4"
  targetNamespace="http://netconfcentral.org/ns/test4"
  elementFormDefault="qualified" attributeFormDefault="unqualified"
  xml:lang="en" version="2009-09-09"
  xmlns:ncx="http://netconfcentral.org/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

      Module: test4
      Version: 2009-09-09
    </xs:documentation>
    <xs:appinfo>
      <ncx:source>
        /home/andy/swdev/yuma/trunk/netconf/modules/test/test4.yang
      </ncx:source>
    </xs:appinfo>
    <xs:appinfo>
      <ncx:revision>
        <ncx:version>2009-09-09</ncx:version>
        <ncx:description>
          Initial revision.
        </ncx:description>
      </ncx:revision>
    </xs:appinfo>
  </xs:annotation>
</xs:schema>
```

Yuma yangdump Manual

```
</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>
            <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>
```

Yuma yangdump Manual

```
<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>
```

2.3.4 SQL TRANSLATION

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

2.3.5 SQL DOCUMENTATION DATABASE TRANSLATION

SQL documentation database translation for the Netconf Central documentation database can be generated using the **--format=sqldb** parameter value. Refer to the Yuma Developer Guide for more details.

2.3.6 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.

Yuma yangdump Manual

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

2.3.7 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

2.3.8 AGENT INSTRUMENTATION H FILE GENERATION

There are 3 different parameters for generating SIL developer H files:

Parameter	Description
--format=h	Generate combined User and Yuma SIL H file
--format=uh	Generate split User SIL H file
--format=yh	Generate split Yuma SIL H file

These parameter values are only relevant for Yuma server developers. Refer to the Yuma Developer Guide for more details.

2.3.9 AGENT INSTRUMENTATION C FILE GENERATION

There are 3 different parameters for generating SIL developer C files:

Parameter	Description
--format=c	Generate combined User and Yuma SIL C file
--format=uc	Generate split User SIL C file

Yuma yangdump Manual

Parameter	Description
--format=yc	Generate split Yuma SIL C file

These parameter values are only relevant for Yuma server developers. Refer to the Yuma Developer Guide for more details.

3 CLI Reference

The **yangdump** program uses command line interface (CLI) parameters to control program behavior. The following sections document all the Yuma CLI parameters relevant to this program, in alphabetical order.

3.1 --config

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

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

--config parameter

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

3.2 --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:	<pre>yangdump \ --defnames=true \ --subtree=~/workdir/ \ --format=html</pre>
----------	--

3.3 --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>

3.4 --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:	<code>yangcli \ --module=test1 \ --deviation=test1_deviations</code>

3.5 --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	empty
Default:	none
Min Allowed:	0

Yuma yangdump Manual

Max Allowed:	1
Supported by:	yangdump
Example:	<pre>yangdump \ --exports \ --module=test3</pre>

3.6 --feature-code-default

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

If 'dynamic', then by default, features can be loaded at run-time, and objects with if-feature statements will be available in case the feature is enabled.

If 'static', then by default, features are set at compile-time, and any disabled objects will be removed at load-time.

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

--feature-code-default parameter

Syntax	enum (dynamic or static)
Default:	dynamic
Min Allowed:	0
Max Allowed:	1
Supported by:	yangdump
Example:	<pre>yangdump --format=c \ --feature-code-default=static \ --module=test</pre>

3.7 --feature-disable

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

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

```
test:feature1
```

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

Parameters for unknown features will be ignored.

--feature-disable parameter

Syntax	formatted string (module:feature)
--------	-----------------------------------

Yuma yangdump Manual

Default:	none
Min Allowed:	0
Max Allowed:	unlimited
Supported by:	yangcli yangdiff yangdump netconfd
Example:	<pre>yangdump --format=c \ --feature-disable=test:feature1 \ --module=test</pre>

3.8 --feature-dynamic

The **--feature-dynamic** parameter controls how **yangdump** will generate dynamic code for a specific feature.

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

```
test:feature1
```

It is an error if a **--feature-static** and **--feature-dynamic** parameter specify the same feature. Parameters for unknown features will be ignored.

--feature-dynamic parameter

Syntax	formatted string (module:feature
Default:	none
Min Allowed:	0
Max Allowed:	unlimited
Supported by:	yangdump
Example:	<pre>yangdump --format=c \ --feature-dynamic=test:feature1 \ --module=test</pre>

3.9 --feature-enable

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

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

```
test:feature1
```

Yuma yangdump Manual

It is an error if a **--feature-disable** and **--feature-enable** parameter specify the same feature. Parameters for unknown features will be ignored.

--feature-enable parameter

Syntax	formatted string (module:feature
Default:	none
Min Allowed:	0
Max Allowed:	unlimited
Supported by:	yangcli yangdiff yangdump netconfd
Example:	<pre>yangdump --format=c \ --feature-enable=test:feature1 \ --module=test</pre>

3.10 --feature-enable-default

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

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

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

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

--feature-enable-default parameter

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

3.11 --feature-static

The **--feature-static** parameter controls how **yangdump** will generate static code for a specific feature.

Yuma yangdump Manual

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

```
test:feature1
```

It is an error if a **--feature-static** and **--feature-dynamic** parameter specify the same feature. Parameters for unknown features will be ignored.

--feature-static parameter

Syntax	formatted string (module:feature
Default:	none
Min Allowed:	0
Max Allowed:	unlimited
Supported by:	yangdump
Example:	<pre>yangdump --format=c \ --feature-static=test:feature1 \ --module=test</pre>

3.12 --format

The **--format** parameter controls the type of **yangdump** conversion desired, if any.

Yuma SIL developers should refer to the **make_sil_dir** script to generate SIL code instead of using the **yangdump** command directly for code generation formats (c, h, uc, uh, yc, yh).

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

Yuma yangdump Manual

- This option is reserved for future use.
- **sqldb**
 - Generate module specific SQL data for the netconfcentral.sql template.
- **h**
 - Generate a module specific **netconfd** agent instrumentation combined SIL H file.
- **c**
 - Generate a module specific **netconfd** agent instrumentation combined SIL C file.
- **uh**
 - Generate a module specific **netconfd** agent instrumentation User SIL H file.
- **uc**
 - Generate a module specific **netconfd** agent instrumentation User SIL C file.
- **yh**
 - Generate a module specific **netconfd** agent instrumentation Yuma SIL H file.
- **yc**
 - Generate a module specific **netconfd** agent instrumentation Yuma SIL C file.

--format parameter

Syntax	enumeration (xsd, sql, sqldb, html, yang, copy, h, c, uc, uh, yc, yh))
Default:	none
Min Allowed:	0
Max Allowed:	1
Supported by:	yangdump
Example:	<pre>yangdump test1 test2 \ --format=xsd \ --defnames=true \ --output=~/.workdir</pre>

3.13 --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

Yuma yangdump Manual

Syntax	empty
Default:	none
Min Allowed:	0
Max Allowed:	1
Supported by:	netconfd yangcli yangdiff yangdump
Example:	<code>yangdump --help</code>

3.14 --help-mode

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

--help-mode parameter

Syntax	choice of 3 empty leafs --brief --normal --full
Default:	normal
Min Allowed:	0
Max Allowed:	1
Supported by:	netconfd yangcli yangdiff yangdump
Example:	<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.

3.15 --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.org/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:	<pre>yangdump \ --html-div=true --format=html --module=test</pre>

3.16 --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

Yuma yangdump Manual

- **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>yangdump \ --html-toc=plain --format=html --module=test</pre>

3.17 --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 'yuma-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
--------	-------

Yuma yangdump Manual

Default:	none
Min Allowed:	0
Max Allowed:	1
Supported by:	yangdump
Example:	<code>yangdump \ --identifiers \ --module=yuma-mysession</code>

3.18 --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:	2
Min Allowed:	0
Max Allowed:	1
Supported by:	netconfd yangcli yangdiff yangdump
Example:	<code>yangdump --indent=4</code>

3.19 --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:	<code>yangdump --log=~/.test.log&</code>

3.20 --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:	<code>yangdump --log-append \ --log=~ /test.log&</code>

3.21 --log-level

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

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

There are 7 settings that can be used:

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

log-level parameter

Syntax	enumeration: off error warn info
--------	--

Yuma yangdump Manual

	debug debug2 debug3 debug4
Default:	--info (--debug for DEBUG builds)
Min Allowed:	0
Max Allowed:	1
Supported by:	netconfd yangcli yangdiff yangdump
Example:	<code>yangdump --log-level=debug \ --log=~/.test.log&</code>

3.22 --modpath

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

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

--modpath parameter

Syntax	string: list of directory specifications
Default:	\$YUMA_MODPATH environment variable
Min Allowed:	0
Max Allowed:	1
Supported by:	netconfd yangcli yangdiff yangdump
Example:	<code>yangdump \ --modpath=~/.testmodules:~/modules: ~/trunk/netconf/modules" \ --module=~/.test42.yang</code>

3.23 --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:	yangcli \ --module=test1 \ --module=test2

3.24 --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:	yangdump \ --modversion \ --module=test

3.25 --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**

Yuma yangdump Manual

- 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>

3.26 **--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:
 - 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.

Yuma yangdump Manual

- 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>

3.27 --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 effect 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
```

Yuma yangdump Manual

```
272 data missing
273 operation not supported
274 operation failed
275 partial operation
```

-show-errors parameter

Syntax	empty
Default:	none
Min Allowed:	0
Max Allowed:	1
Supported by:	yangdump
Example:	<code>yangdump --show-errors</code>

3.28 --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>

3.29 --stats

The **--stats** parameter is used to request a YANG usage statistics report.

See the **--totals** parameter to control statistics totals reporting, when used with this parameter.

Yuma yangdump Manual

Statistics are not normally collected. This must be enabled by setting this parameter to a value other than 'none' (the default). Any other value will cause statistics to be collected and reported after each input module is validated.

The verbosity of the statistics report is controlled with this parameter. Each enumeration includes all the previous statistics (from lower enumerations), plus additional statistics. The following table describes the counters that are displayed at each minimum **--stats** enumeration value.

counter	level	description
Complexity score	brief	Numeric score characterizing the level of implementation complexity for the module. A higher number indicates a more complex module.
Total nodes	brief	The total number of object nodes (data, notification, rpc) in the module. <ul style="list-style-type: none"> The Config nodes + Non-config nodes should equal this counter. The Mandatory nodes + Optional nodes should equal this counter.
Extensions	basic	The number of extension statements.
Features	basic	The number of feature statements.
Groupings	basic	The number of exported groupings.
Typedefs	basic	The number of exported typedefs.
Deviations	basic	The number of deviation statements.
Top Data Nodes	basic	The number of top-level data nodes.
Config nodes	basic	The number of configurable nodes.
Non-config nodes	basic	The number of non-configurable nodes.
Mandatory nodes	advanced	The number of mandatory nodes.
Optional nodes	advanced	The number of optional nodes.
Notifications	advanced	The number of notification statements.
Rpcs	advanced	The number of rpc statements.
Rpc inputs	advanced	The number of RPC input statements.
Rpc outputs	advanced	The number of rpc output statements
Augments	advanced	The number of top-level augment statements within data nodes (not groupings).
Uses	advanced	The number of uses statements within data nodes (not groupings).
Choices	advanced	The number of choice statements.
Cases	advanced	The number of case statements. This includes implied cases which contain a single data node.
Anyxmls	advanced	The number of anyxml statements.
NP containers	advanced	The number of non-presence containers.

Yuma yangdump Manual

counter	level	description
P containers	advanced	The number of presence containers.
Lists	advanced	The number of list statements.
Leaf-lists	advanced	The number of leaf-list statements.
Key leafs	advanced	The number of leaf statements which are defined within a list to be a key.
Plain leafs	advanced	The number of plain leaf statements.
Imports	all	The number of import statements.
Integral numbers	advanced	The number of leaf and leaf-list statements that used a numeric data type other than decimal64.
Decimal64s	advanced	The number of leaf and leaf-list statements that used the decimal64 data type.
Enumerations	advanced	The number of leaf and leaf-list statements that used the enumeration data type.
Bits	advanced	The number of leaf and leaf-list statements that used the bits data type.
Booleans	advanced	The number of leaf and leaf-list statements that used the boolean data type.
Emptys	advanced	The number of leaf and leaf-list statements that used the empty data type.
Strings	advanced	The number of leaf and leaf-list statements that used the string data type.
Binarys	advanced	The number of leaf and leaf-list statements that used the binary data type.
Instance Identifiers	advanced	The number of leaf and leaf-list statements that used the instance-identifier data type.
Leafrefs	advanced	The number of leaf and leaf-list statements that used the leafref data type.
Identityrefs	advanced	The number of leaf and leaf-list statements that used the identityref data type.
Unions	advanced	The number of leaf and leaf-list statements that used the union data type.

--stats parameter

Yuma yangdump Manual

Syntax	enumeration [none, brief, basic, advanced, all]
Default:	none
Min Allowed:	0
Max Allowed:	1
Supported by:	yangdump
Example:	<pre>yangdump \ --stats=advanced \ --module=test</pre>

3.30 --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:	<pre>yangdump \ --subdirs=false \ --subtree=/testpath</pre>

3.31 --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:

Yuma yangdump Manual

- 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>

3.32 --totals

The **--totals** parameter is used with the **--stats** parameter to control how summary statistics are reported.

- The **--stats** parameter must be set to a value other than 'none' for this parameter to have any affect. The value of this parameter will control which statistics are reported.
- Normally, no summary is generated (default is 'none').
- The 'summary' value will not have any affect unless there is more than one input module in the statistics collection.
- The 'summary-only' value will cause the module statistics reports to be skipped, and only a summary of all the input modules will be displayed.

--totals parameter

Syntax	enumeration [none, summary, summary-only]
Default:	none
Min Allowed:	0
Max Allowed:	1
Supported by:	yangdump

Example:	<pre>yangdump \ ---stats=basic --totals=summary-only --subtree=~ /my-test-modules</pre>
----------	---

3.33 --tree-identifiers

The **--tree-identifiers** parameter causes information to be object identifier name 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)
- local name for the object (indented for each descendant node)

Example report for module 'yuma-mysession':

```

identifiers:
  rpc get-my-session
    container output
      leaf indent
      leaf linesize
      leaf with-defaults
    container input
  rpc set-my-session
    container input
      leaf indent
      leaf linesize
      leaf with-defaults
    container output
```

--tree-identifiers parameter

Syntax	empty
Default:	none
Min Allowed:	0
Max Allowed:	1
Supported by:	yangdump
Example:	<pre>yangdump \ --tree-identifiers \ --module=yuma-mysession</pre>

3.34 --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.

Yuma yangdump Manual

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>

3.35 --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):

--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>

3.36 --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:

DEBUG: <major>.<minor>.<svn-build-version>

or

NON-DEBUG: <major>.<minor>-<release>

An example version number that may be printed:

yangdump 2.0-0

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

3.37 --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>

3.38 --warn-idlen

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

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

--warn-idlen parameter

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

3.39 --warn-linelen

Yuma yangdump Manual

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

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

Tab characters are counted as 8 spaces.

--warn-linelen parameter

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

3.40 --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>yangdump --warn-off=435</code> <code># revision order not descending</code>

3.41 --xsd-schemaloc

The **--xsd-schemaloc** parameter specifies that **yangdump** XSD translations should include the 'schemaLocation' attribute.

Yuma yangdump 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.org/ns/test3
http://example.com/public/test3.xsd'
```

- if **--versionnames=true** (default):

```
xsi:schemaLocation='http://netconfcentral.org/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:	yangdump \ --xsd-schemaloc="http://example.com" --format=xsd --module=test3 --defnames=true

3.42 --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 yangdump Manual

- **\$YUMA_HOME/data**
 - This directory is searched for data files.
- **\$YUMA_HOME/scripts**
 - This directory is searched for **yangcli** script files.

yuma-home parameter

Syntax	string: directory specification
Default:	\$YUMA_HOME environment variable
Min Allowed:	0
Max Allowed:	1
Supported by:	netconfd yangcli yangdiff yangdump
Example:	<pre>yangdump \ --yuma-home=~ /sw/netconf \ --log=~ /server.log&</pre>