NETCONF

In This Chapter

This chapter provides information to configure NETCONF.

Topics in this chapter include:

NETCONF Overview

NETCONF Introduction

NETCONF is a standardized IETF configuration management protocol published in RFC 6241. It is secure, connection oriented, and runs on top of the SSHv2 transport protocol as specified in RFC 6242. NETCONF can be used as an alternative to CLI or SNMP for managing an SR OS node.

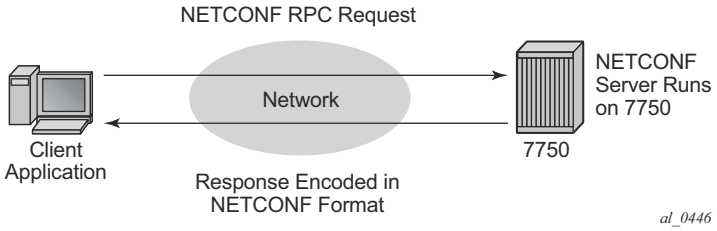

Figure 10: NETCONF RPC Request

NETCONF is an XML-based protocol used to configure network devices. It uses RPC messaging for communication between a NETCONF client and the NETCONF server running on the SR OS node. An RPC message and configuration data is encapsulated within an XML document. These XML documents are exchanged between a NETCONF client and a NETCONF server in a request/response type of interaction. The SR OS NETCONF interface supports both configuration support and retrieval of operational information.

NETCONF can be conceptually partitioned into four layers as described in RFC 6241.

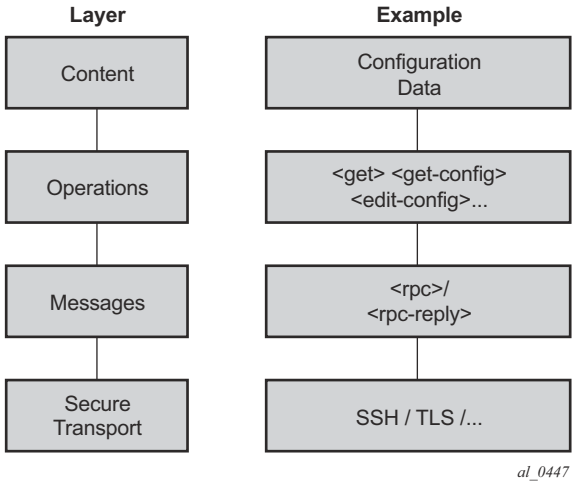

Figure 11: NETCONF Layers (RFC 6241)

NETCONF in SR OS

NETCONF can be used on an SR OS router to perform router management operations including:

- Change the configuration of the router (<edit-config> operation)
- Read the configuration of the router (<get-config> operation, equivalent to the "info" command in CLI)
- Read operational status and data (and associated configuration information) (<get> operation, equivalent to the "show" commands in CLI)

NETCONF is not used for notifications on an SR OS router; for example, log events, syslog, or SNMP notifications (traps).

The equivalent of some admin commands are available via the SR OS NETCONF interface:

- "admin save" can be done using the <copy-config> operation.
- "admin rollback" commands are supported using a CLI content layer <cli-action> RPC.

“bof”, “debug”, “tools”, and other general CLI operational commands (e.g. “telnet” or “ping”) are not supported via NETCONF on an SR OS router.

The SR OS NETCONF server advertises base capability 1.1 (in addition to 1.0).

SR OS supports both a CLI content layer and an XML-based content layer for NETCONF.

YANG Data Models

The SR OS NETCONF XML content layer configuration schema is described in a set of Alcatel-Lucent proprietary YANG modules. The configuration modules are advertised in the SR OS NETCONF server hello.

The configuration YANG data model closely aligns to the SR OS CLI configuration tree structure and commands.

A set of YANG modules are published and distributed as part of an SR OS image in the cflash/support directory (along with files like dictionary-freeradius.txt and stats.dtd).

The following areas of CLI do not have equivalent YANG data models:

- **bof**
- **admin**, **tools**, **debug**, or **show** branches

Transport and Sessions

SSH transport is supported on TCP port 830 with IPv4 or IPv6 in the Base routing instance. NETCONF SSH sessions (like CLI, SCP and sFTP sessions) are subject to any configurable and non-configurable session limits; for example, inbound-max-sessions. Both the SSH server and NETCONF protocol must be enabled in the router configuration in order to use NETCONF. NETCONF sessions can be disconnected using the "admin disconnect" command.

NETCONF sessions do not time out automatically and are not subject to the CLI session timeout. Operators can disconnect sessions manually if they need to.

A client establishing a NETCONF session must log into the router so user accounts must exist for NETCONF on the SR. A new access type 'netconf' is provided. The user must be configured with both 'console' and 'netconf' access.

Only authentication via the local user database is supported for NETCONF users/sessions (no RADIUS or TACACS+ authentication). Access to various CLI config and show commands (authorization) via NETCONF is controlled through the profile assigned to the user that is used to authenticate the underlying SSH session.

Access to LI commands is based on the “access li” setting for the user.

If a NETCONF request attempts to execute a CLI command which is outside the scope of its access profile, an error response will be sent. For example:

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
        <get>
```

```
                <filter>
                        <oper-data-format-cli-block>
                                <cli-show>system security</cli-show>
                        </oper-data-format-cli-block>
                </filter>
        </get>
</rpc>
]]>]]>
```

Reply:

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <rpc-error>
        <error-type>application</error-type>
        <error-tag>operation-failed</error-tag>
        <error-severity>error</error-severity>
        <error-info>
            <err-element>cli-show</err-element>
        </error-info>
        <error-message>
            command failed - 'show system security'
            MINOR: CLI Command not allowed for this user.
        </error-message>
    </rpc-error>
</rpc-reply>
]]>]]>
```

NETCONF Operations

The following base protocol operations are supported:

- <get>
- <get-config>
- <edit-config>
- <copy-config>
- <delete-config>
- <validate>
- <close-session>
- <kill-session>

The <lock> and <unlock> base protocol operations are not supported.

The <error-option> is not supported. SR OS implements the stop-on-error behavior by default. The continue-on-error and rollback-on-error are not supported.

<get>

CLI content layer <get> operation is supported. XML content layer <get> operation is not supported.

A <get> request is first analyzed for syntax errors before any execution starts. If a syntax error is found then a single global <rpc-error> for the entire request is sent in the reply.

Responses are provided for each item in the request until the first item with an error is found. The item with an error has a <response> tag containing some error information, followed by an <rpc-error> tag (and sub-tags). The reply is then returned and subsequent items are not executed.

The <rpc-error> for an individual item (i.e. for a non-syntax error) is after the </response> information and not inside the <response>.

Example — <get> request with a non-syntax error in the 2nd item:

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
        <get>
                <filter>
                        <oper-data-format-cli-block>
                                <cli-show>router interface "system"</cli-show>
                                <cli-show>router mpls lsp</cli-show>
                                <cli-show>system security ssh</cli-show>
                        </oper-data-format-cli-block>
                </filter>
        </get>
</rpc>
]]>]]>
```

Reply:

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <data xmlns="urn:alcatel-lucent.com:sros:ns:yang:cli-content-layer-r13">
        <oper-data-format-cli-block>
            <item>
                <cli-show>router interface "system"</cli-show>
                <response>

===============================================================================
Interface Table (Router: Base)
===============================================================================
Interface-Name                   Adm         Opr(v4/v6)  Mode    Port/SapId
   IP-Address                                                    PfxState
-------------------------------------------------------------------------------
system                           Up          Up/Down     Network system
   144.23.63.5/32                                                n/a
-------------------------------------------------------------------------------
Interfaces : 1
===============================================================================
                </response>
```

```
                </item>
                <item>
                    <cli-show>router mpls lsp</cli-show>
                    <response>
                        MINOR: CLI MPLS is not configured.
                    </response>
                    <rpc-error>
                        <error-type>application</error-type>
                        <error-tag>operation-failed</error-tag>
                        <error-severity>error</error-severity>
                        <error-info>
                            <err-element>cli-show</err-element>
                        </error-info>
                        <error-message>
                            command failed - 'show router mpls lsp'
                        </error-message>
                    </rpc-error>
                </item>
            </oper-data-format-cli-block>
    </data>
</rpc-reply>
]]>]]>
```

<get-config>

<get-config> returns non-default configuration by default (i.e. the 'trim' mode as per RFC 6243).

<edit-config>

The following values for the <test-option> parameter under <edit-config> are supported:

- test-then-set
- set
- test-only

<copy-config> and <delete-config>

The <copy-config> and <delete-config> base protocol operations are supported for specific combinations of source and target datastores.

The <copy-config> operation is supported for the following combinations of sources and targets:

- <source>=<url> and <target>=<startup> (as long as both are not remote urls)
- <source>=<startup> and <target>=<url> (as long as both are not remote urls)
- <source>=<running> and <target>=<url>
  - → Equivalent of "admin save <file-url>"
  - → An index file is also saved if "persist on" is configured in the bof

- <source>=<running> and <target>=<startup>
  - → Equivalent of "admin save"
  - → An index file is also saved if "persist on" is configured in the bof

<running> cannot be a <target> for a <copy-config>.

Remote url to remote url copies are not supported. For example, if primary-image is a remote url then a <startup> to copy will fail with an error.

The <copy-config> operation uses the CLI Content Layer format. The format of the source and target is block CLI.

The <delete-config> operation is supported for the following targets:

- <url>
- <startup>

<delete-config> is not allowed on the <running> datastore.

<validate>

The validate:1.1 capability is supported:

- The validate:1.1 and 1.0 capabilities are advertised in the NETCONF server's <hello>:
  - → <capability>urn:ietf:params:netconf:capability:validate:1.0</capability>
  - → <capability>urn:ietf:params:netconf:capability:validate:1.1</capability>
- The <validate> request is supported for an XML content layer request but not for a CLI content layer request. Detection of a <config-format-cli-block> or <oper-data-format-cli-block> tag in a <validate> request will result in an "operation not supported" error response.
- A <validate> request is supported for a selection of config (<source><config>), or for the <running> datastore, which only returns 'OK'. <validate> is not supported for url sources or the <startup> datastore.

Datastores and URLs

SR OS supports the <running> datastore, the <startup> datastore, and <url> tags (**Note:** <url> is not a datastore in itself). The <candidate> datastore is not supported.

All configuration changes (<edit-config>) done to the <running> datastore via NETCONF take immediate operational effect.

The <startup> datastore and <url> tags can only be used with <copy-config> and <delete-config> and are not supported with any other operations (including <edit-config>, <get-config>, <get>, <validate>, etc).

The :startup capability is advertised in the SR OS NETCONF server <hello>:

```
<capability>urn:ietf:params:netconf:capability:startup:1.0</capability>
```

url supports the same options as CLI <file-url>: local urls (CF) and remote urls (ftp and tftp).

The :url capability is advertised in the SR OS NETCONF server <hello>:

```
<capability>urn:ietf:params:netconf:capability:url:1.0?scheme=ftp,tftp,file</
capability>
```

The following examples show the format of each URL scheme (Note the "///" for the 'file' URL. The 'file://localhost/...' format is not supported.):

- <target><url>ftp://name:passwd@a.b.c.d/usr/myfiles/myfile.cfg</url></target>
- <target><url>tftp://name:passwd@a.b.c.d/usr/myfiles/myfile.cfg</url></target>
- <target><url>file:///cf3:/myfiles/myfile.cfg</url></target>
- **Note:** The following format is also supported (no 'file:///'): <target><url>cf3:/myfiles/myfile.cfg</url></target>

The <startup> datastore is identified by following the bof primary-config/secondary-config/tertiary-config paths as configured by the operator. <startup> is effectively an alias for a url (a special url used for system startup) with some extra resiliency (primary/secondary/tertiary).

The bof is not considered as part of any config datastore.

Debug config (such as debug mirrors, or anything saved with "admin debug-save") is not considered as part of any config datastore.

Lawful Interception configuration information is contained in the <running> datastore but is not saved in the <startup> datastore. The equivalent of the CLI "li save" command is available in an <edit-config>.

Configuration changes done via NETCONF are subject to CLI Rollback (revert, save, etc) and are included in the configuration when the operator performs an "admin save" in CLI.

General NETCONF behavior

Pressing Ctrl-C in a NETCONF session will immediately terminate the session.

The SR OS NETCONF implementation does not support XML namespaces (xmlns). Any XML namespace or prefix declarations in the rpc tag are accepted and returned in the rpc-

reply tag but ignored and unused. Any XML namespace or prefix declarations in the rest of the request are ignored and unused. The SR OS NETCONF server puts the correct NETCONF namespace declaration (“urn:ietf:params:xml:ns:netconf:base:1.0”) in all replies.

Example 1 — The standard NETCONF namespace “urn:ietf:params:xml:ns:netconf:base:1.0” defined more than once in the rpc tag:

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc message-id="101"
xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns:alu="urn:ietf:params:xml:ns:netconf:base:1.0">
<get-config>
<source> <running/> </source>
<filter>
    <configure>
        <router>
            <interface>
               <interface-name>"system"</interface-name>
            </interface>
        </router>
    </configure>
</filter>
</get-config>
</rpc>
]]>]]>
```

Reply (no error message):

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply message-id="101" xmlns:alu="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <data>
        <configure xmlns="urn:alcatel-lucent.com:sros:ns:yang:conf-r13">
            <router>
                <router-name>Base</router-name>
                <interface>
                    <interface-name>system</interface-name>
                    <address>
                        <ip-address-mask>144.23.63.5/32</ip-address-mask>
                    </address>
                    <shutdown>false</shutdown>
                </interface>
            </router>
        </configure>
    </data>
</rpc-reply>
]]>]]>
```

Example 2 — A non-NETCONF base namespace defined in the rpc tag:

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc message-id="101"
xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns:alu="urn:alcatel-lucent.com:sros:ns:yang:conf-r13">
<get-config>
<source> <running/> </source>
```

```
<filter>
    <configure xmlns="urn:alcatel-lucent.com:sros:ns:yang:conf-r13">
        <router>
            <interface>
              <interface-name>"system"</interface-name>
            </interface>
        </router>
    </configure>
</filter>
</get-config>
</rpc>
]]>]]>
```

Reply (non-standard namespace used in rpc tag is ignored):

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns:alu="urn:alcatel-lucent.com:sros:ns:yang:conf-r13">
<data>
     <configure xmlns="urn:alcatel-lucent.com:sros:ns:yang:conf-r13">
          <router>
               <router-name>Base</router-name>
               <interface>
                    <interface-name>system</interface-name>
                    <address>
                         <ip-address-mask>144.23.63.5/32</ip-address-mask>
                    </address>
                    <shutdown>false</shutdown>
               </interface>
          </router>
     </configure>
</data>
</rpc-reply>
]]>]]>
```

Example 3 — A non-standard NETCONF namespace used in one of the tags but not defined in the rpc tag:

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc message-id="101"
xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns:alu="urn:ietf:params:xml:ns:netconf:base:1.0">
<get-config>
<source> <running/> </source>
<filter>
    <configure>
        <router>
            <interface xmlns:alu="urn:alcatel-lucent.com:sros:ns:yang:conf-r13">
                 <interface-name>"system"</interface-name>
            </interface>
        </router>
    </configure>
</filter>
</get-config>
</rpc>
]]>]]>
```

Reply (non-standard namespace used in tag is ignored):

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply message-id="101" xmlns:alu="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <data>
        <configure xmlns="urn:alcatel-lucent.com:sros:ns:yang:conf-r13">
            <router>
                <router-name>Base</router-name>
                <interface>
                    <interface-name>system</interface-name>
                    <address>
                        <ip-address-mask>144.23.63.5/32</ip-address-mask>
                    </address>
                    <shutdown>false</shutdown>
                </interface>
            </router>
        </configure>
    </data>
</rpc-reply>
]]>]]>
```

Example 4 — A non-standard NETCONF namespace/prefix used in one of the tags but not defined in the rpc tag:

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc message-id="101"
xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns:alu="urn:ietf:params:xml:ns:netconf:base:1.0">
<get-config>
<source> <running/> </source>
<filter>
    <configure>
        <router>
            <interface xmlns:alu="urn:alcatel-lucent.com:sros:ns:yang:conf-r13">
                 <alu:interface-name>"system"</alu:interface-name>
            </interface>
        </router>
    </configure>
</filter>
</get-config>
</rpc>
]]>]]>
```

Reply (non-standard namespace/prefix used in tag is ignored):

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply message-id="101" xmlns:alu="urn:ietf:params:xml:ns:netconf:base:1.0"
xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <data>
        <configure xmlns="urn:alcatel-lucent.com:sros:ns:yang:conf-r13">
            <router>
                <router-name>Base</router-name>
                <interface>
                    <interface-name>system</interface-name>
                    <address>
                        <ip-address-mask>144.23.63.5/32</ip-address-mask>
```

```
                    </address>
                    <shutdown>false</shutdown>
                </interface>
            </router>
        </configure>
    </data>
</rpc-reply>
]]>]]>
```

The chunked framing mechanism is supported (in addition to the EOM mechanism). As per RFC 6242, Section 4.1 - Framing Protocol, "... If the :base:1.1 capability is advertised by both peers, the chunked framing mechanism (see Section 4.2) is used for the remainder of the NETCONF session. Otherwise, the old end-of-message-based mechanism (see Section 4.3) is used."

Example 1 — Chunked message:

```
#302
<?xml version="1.0" encoding="UTF-8"?><rpc message-id="101"
xmlns="urn:ietf:params:xml:ns:netconf:base:1.0"><get-config><source><running/></
source><filter><config><configure><router><interface><interface-name>system</inter-
face-name></interface></router></configure></config></filter></get-config></rpc>
##
```

Example 2 — Chunked message:

```
#38
<?xml version="1.0" encoding="UTF-8"?>
#85
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <get-config>
#62
    <source><running/></source>
    <filter>
      <configure>
##79
        <system>
            <netconf>
            </netconf>
        </system>
##55
      </configure>
    </filter>
  </get-config>
</rpc>
##
```

Handling of default data (for example, ' info' vs 'info detail') is done using the mechanisms detailed in RFC 6243. The SR OS NETCONF server supports the 'trim' method and advertises that in the <hello>:

```
<capability>urn:ietf:params:netconf:capability:with-defaults:1.0?basic-mode=trim</
capability>
```

Pseudo-transactional capabilities are supported. A user can save a rollback checkpoint (for example, prior to doing an <edit-config> or a series of <edit-config>) and perform a rollback revert if needed later.

Example 1 — Two rollback items with responses:

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc message-id="102" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <cli-action>
    <admin>rollback compare active-cfg to 1</admin>
    <admin>rollback compare</admin>
  </cli-action>
</rpc>
]]>]]>
```

Reply:

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply message-id="102" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <data xmlns="urn:alcatel-lucent.com:sros:ns:yang:cli-content-layer-r13">
        <cli-action>
            <item>
                <admin>rollback compare active-cfg to 1</admin>
         <response>
0.150 s
0.450 s
-----------------------------------------------
  configure
     router
-        mpls
-            shutdown
-            interface "system"
-                no shutdown
-            exit
-            lsp "test"
-                shutdown
-            exit
-        exit
-        rsvp
-            shutdown
-            interface "system"
-                no shutdown
-            exit
-        exit
     exit
  exit
-----------------------------------------------
Finished in 0.720 s
               </response>
            </item>
            <item>
                <admin>rollback compare</admin>
            <response>
0.160 s
0.070 s
-----------------------------------------------
  configure
```

```
     router
-        mpls
-            shutdown
-            interface "system"
-                no shutdown
-            exit
-            lsp "test"
-                shutdown
-            exit
-        exit
-        rsvp
-            shutdown
-            interface "system"
-                no shutdown
-            exit
-        exit
     exit
     service
-        vpls "99" customer 1 create
-            shutdown
-            stp
-                shutdown
-            exit
-        exit
     exit
  exit
-------------------------------------------------
Finished in 0.350 s
               </response>
           </item>
       </cli-action>
    </data>
</rpc-reply>
]]>]]>
```

Example 2 — Syntax error in the request resulting in global rpc-error reply:

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc message-id="103"
    xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <cli-action>
      <admin>rollback compare active-cfg to 1</admin>
      <admin>rollback compare flee-fly</admin>
  </cli-action>
</rpc>
]]>]]>
```

Reply:

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply message-id="103" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <rpc-error>
        <error-type>application</error-type>
        <error-tag>operation-failed</error-tag>
        <error-severity>error</error-severity>
        <error-info>
            <err-element>admin</err-element>
```

```
        </error-info>
        <error-message>
            command failed - '/admin rollback compare flee-fly'
        </error-message>
    </rpc-error>
</rpc-reply>
]]>]]>
```

Example 3 — Error processing the request:

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc message-id="103"
    xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <cli-action>
      <admin>rollback compare active-cfg to 1</admin>
      <admin>rollback compare 1 to flee-fly</admin>
  </cli-action>
</rpc>
]]>]]>
```

Reply:

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply message-id="103" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <data xmlns="urn:alcatel-lucent.com:sros:ns:yang:cli-content-layer-r13">
        <cli-action>
            <item>
                <admin>rollback compare active-cfg to 1</admin>
           <response>
0.160 s
0.180 s
-----------------------------------------------
  configure
     router
-        mpls
-            shutdown
-            interface "system"
-                no shutdown
-            exit
-        exit
-        rsvp
-            shutdown
-            interface "system"
-                no shutdown
-            exit
-        exit
     exit
  exit
-----------------------------------------------
Finished in 0.460 s
                </response>
            </item>
            <item>
                <admin>rollback compare 1 to flee-fly</admin>
                <response>
                </response>
                <rpc-error>
```

```
                    <error-type>application</error-type>
                    <error-tag>operation-failed</error-tag>
                    <error-severity>error</error-severity>
                    <error-info>
                        <err-element>admin</err-element>
                    </error-info>
                    <error-message>
                        command failed - '/admin rollback compare 1 to flee-fly'
                        MINOR: CLI No such file ('flee-fly').
                    </error-message>
                </rpc-error>
            </item>
        </cli-action>
    </data>
</rpc-reply>
]]>]]>
```

Example 4 — Error in the 2nd item of the request, resulting in no 3rd item in the reply:

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc message-id="104" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <cli-action>
    <admin>rollback compare active-cfg to 1</admin>
    <admin>rollback compare 1 to xyz</admin>
    <admin>rollback compare active-cfg to 1</admin>
  </cli-action>
</rpc>
]]>]]>
```

Reply:

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply message-id="104" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <data xmlns="urn:alcatel-lucent.com:sros:ns:yang:cli-content-layer-r13">
        <cli-action>
            <item>
                <admin>rollback compare active-cfg to 1</admin>
          <response>
0.170 s
1.350 s
-----------------------------------------------
  configure
     router
-        mpls
-            shutdown
-            interface "system"
-                no shutdown
-            exit
-        exit
-        rsvp
-            shutdown
-            interface "system"
-                no shutdown
-            exit
-        exit
     exit
  exit
```

```
-----------------------------------------------
Finished in 1.640 s
                </response>
            </item>
            <item>
                <admin>rollback compare 1 to xyz</admin>
                <response>
                </response>
                <rpc-error>
                    <error-type>application</error-type>
                    <error-tag>operation-failed</error-tag>
                    <error-severity>error</error-severity>
                    <error-info>
                        <err-element>admin</err-element>
                    </error-info>
                    <error-message>
                        command failed - '/admin rollback compare 1 to xyz'
                        MINOR: CLI No such file ('xyz').
                    </error-message>
                </rpc-error>
            </item>
        </cli-action>
    </data>
</rpc-reply>
]]>]]>
```

System Provisioned Configuration (SPC) Objects

There are a set of configuration objects that are provisioned (added to the running datastore) automatically by SR OS; for example, log-id 99.

Some of these items can be deleted/removed by a user (Deletable SPC Objects):

- In CLI these are removed by specifying the keyword 'no' which is then visible in an "info" command or in a saved config (admin save); for example, 'no log-id 99'.
- The Deletable SPC Objects can be removed or re-created via NETCONF <edit-config> requests, but they are not visible in a <get-config> response when they are:
  - → Set to their default values (including all child leaves & objects)
  - → Removed or deleted
- The Deletable SPC Objects are visible in a <get-config> response if a child leaf or object is changed away from the default value; for example, changing log-99 to time-format local.
- The list of Deletable SPC Objects is as follows:

```
Config system security profile default
Config system security profile default entry 10-100
Config system security profile administrative
Config system security profile administrative entry 10-112
Config system security user "admin"
Config system security user console member "default"
```

```
Config system security snmp access group  xyz (a set of access groups)
Config system security ssh client-cipher-list protocol-version 1 cipher 200-210
Config system security ssh client-cipher-list protocol-version 2 cipher 190-235
Config system security ssh server-cipher-list protocol-version 1 cipher 200-205
Config system security ssh server-cipher-list protocol-version 2 cipher 190-235
Config log filter 1001
Config log filter 1001 entry 10
Config log log-id 99 & 100
```

Some SPC objects cannot be deleted (Non-Deletable SPC Objects):

- Although these objects cannot be deleted, some of them contain leafs that can be modified.
- The Non-Deletable SPC Objects are not visible in a <get-config> response when they are set to their default values (including all child leaves & objects).
- The Non-Deletable SPC Objects are visible in a <get-config> response if a child leaf or object is changed away from the default value; for example, setting the card-type.
- The list of Non-Deletable SPC Objects is as follows:

```
Config system security user-template {tacplus_default|radius_default}
Config system security snmp view iso …
Config system security snmp view li-view …
Config system security snmp view mgmt-view …
Config system security snmp view vprn-view …
Config system security snmp view no-security-view …Config log event-control …
Config filter log 101
Config qos … various default policies can’t be deleted
Config qos queue-group-templates … these can’t be deleted
Config card <x>
Config router network-domains network-domain “default”
Config oam-pm bin-group 1
Config call-trace trace-profile “default”
```

There are some Non-Deletable SPC Objects that are visible in a <get-config> request even if they are set to default values:

```
Config system security cpu-protection policy 254 and 255
Config router interface “system"
Config service customer 1
```

Establishing a NETCONF Session

The following example shows a client on a Linux PC initiating a connection to an SR OS NETCONF server. The SSH session must be invoked using an SSH subsystem (as recommended in RFC 6242):

```
ssh -s my_username@a.b.c.d -p 830 netconf
```

The following example shows an exchange of hello messages which include advertisement of capabilities.

From the SR OS server:

```
<?xml version="1.0" encoding="UTF-8"?>
<hello xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <capabilities>
        <capability>urn:ietf:params:netconf:base:1.0</capability>
        <capability>urn:ietf:params:netconf:base:1.1</capability>
        <capability>urn:ietf:params:netconf:capability:writable-running:1.0</capabil-
ity>
        <capability>urn:ietf:params:netconf:capability:validate:1.0</capability>
        <capability>urn:ietf:params:netconf:capability:validate:1.1</capability>
        <capability>urn:ietf:params:netconf:capability:startup:1.0</capability>
        <capability>urn:ietf:params:netconf:capability:url:1.0?scheme=ftp,tftp,file</
capability>
        <capability>urn:ietf:params:netconf:capability:with-defaults:1.0?basic-
mode=trim</capability>
        <capability>urn:ietf:params:xml:ns:netconf:base:1.0?module=ietf-net-
conf&amp;revision=2015-02-27&amp;features=writable-running,vali-
date,startup,url&amp;deviations=alu-netconf-deviations-r13</capability>
        <capability>urn:alcatel-lucent.com:sros:ns:yang:netconf-deviations-r13?mod-
ule=alu-netconf-deviations-r13&amp;revision=2015-02-27</capability>
        <capability>urn:alcatel-lucent.com:sros:ns:yang:cli-content-layer-r13?mod-
ule=alu-cli-content-layer-r13&amp;revision=2015-02-27</capability>
        <capability>urn:alcatel-lucent.com:sros:ns:yang:conf-r13?module=conf-
r13&amp;revision=2015-02-27</capability>
        <capability>urn:alcatel-lucent.com:sros:ns:yang:conf-aaa-r13?module=conf-aaa-
r13&amp;revision=2015-02-27</capability>
     ...
     ...
     ...
     ...
        <capability>urn:alcatel-lucent.com:sros:ns:yang:conf-vsm-r13?module=conf-vsm-
r13&amp;revision=2015-02-27</capability>
    </capabilities>
    <session-id>54</session-id>
</hello>
]]>]]>
```

From a client:

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

XML Content Layer

XML is the default content layer format for the SR OS NETCONF server. When using the XML format at the NETCONF content layer, configuration changes and configuration information retrieved are expressed as XML tags.

The XML formatted configuration information must be correctly ordered and has the same dependencies and behavior as the equivalent CLI commands.

<edit-config> with XML Content Layer

An <edit-config> operation is supported with the <running> datastore only. The following <edit-config> operation attribute values are supported:

- merge
- remove
- delete
  - → A 'delete' operation for a leaf or a presence container will not return an error if the item is already deleted.
  - → An error is returned if attempting to delete a list node that doesn't exist.
  - → A 'delete' operation for a container without presence will return an error
- create
  - → A 'create' operation for a leaf or a presence container will not return an error if the item is being set to the same value.
  - → An error is returned if attempting to create a list node that already exists.
  - → A 'create' operation for a container without presence will result in an "OK" response (no error) but will be silently ignored.

'replace' is not supported as an attribute value for the <edit-config> operation.

Both 'delete' and 'remove' operations have the following behavior:

- Delete or remove operations are not supported for boolean leafs. For example, any of the following samples will return an error:
  - → <shutdown operation="delete"/>
  - → <shutdown operation="delete">false</shutdown>
  - → <interface operation="delete">

    <interface-name>abc</interface-name>

    <shutdown>true</shutdown>

    </interface>

    (For this last case <shutdown operation="merge">true</shutdown> could be used instead to make the request valid.)
- A delete or remove operation is the equivalent of 'no xyz' in CLI. This 'no xyz' is applied whether the default for xyz is enabled ('xyz'), disabled ('no xyz') or some specific value. The delete operation is not aware of the default value of the object/leaf being deleted.

- A delete or remove for a leaf, where the request also specifies a value for the leaf, will result in an error.

The <edit-config> <default-operation> parameter is supported with the following values: merge, none. The 'replace' value is not supported. An operation of "none" on a leaf node (inherited or direct) causes that leaf statement to be ignored. No error will be returned if the leaf does not exist in the data model.

For 'merge' and 'create' operations the operations and tags specified in an <edit-config> request are order-aware and order-dependant and the sequence of operations must follow the required sequence of the equivalent CLI commands. The <edit-config> is processed and executed in a top-down order. The same leaf can be enabled, disabled, enabled and then disabled and the final result is whatever was last specified for that leaf in the <edit-config> request.

For 'delete' and 'remove' operations the SR OS NETCONF server will recursively "unwind" any children of the node being deleted or removed first before removing the node itself. The 'deepest' child branch of the request is examined first and any leafs are processed, after which the server works backwards out of the deepest branches back up to the object where the delete operation was specified. Note that if children branches of an object are required to be removed before deleting the object in CLI, then the equivalent delete request in a NETCONF <edit-config> must contain all those children if they exist, such as if the children are configured in the config datastore). For example:

```
<config>
    <configure>
        <service>
            <vpls operation="delete">
                <service-id>11</service-id>
                <interface>
                    <ip-int-name>test</ip-int-name>
                    <shutdown operation="merge">true</shutdown>
                </interface>
                <shutdown operation="merge">true</shutdown>
            </vpls>
        </service>
    </configure>
</config>
```

In the example above, SR OS will first shutdown the test interface, then delete the interface, then shutdown the VPLS and then finally remove it.

Note that the 'operation="merge"' is required in the shutdown nodes because otherwise the inherited operation is delete which is not supported on boolean leafs.

If other children of vpls 11 exist in the config besides the interface 'test' specified in the delete request above, and those children are required in CLI to be deleted before removing vpls 11, then the deletion request above will fail. All configured children must be specified in the delete request.

<get-config> with XML Content Layer

A <get-config> operation is supported with the <running> datastore only.

Subtree filtering for basic subtree selection is supported for XML content layer <get-config> requests. Post-filtering of the selected subtrees is not supported. The details of subtree filter support are as follows:

- Attribute match expressions (section 6.2.2 of RFC 6241) are not supported. See details below about content match nodes.
- Only containers are supported as selection nodes (section 6.2.4 of RFC 6241). Empty leaf nodes or list name nodes are not supported as selection nodes.
  - → Nodes that represent lists must also include content match nodes for all keys of the list; for example, <configure><router><interface><interface-name>abc</interface-name>.
  - → A selection node that is a list, without also specifying the key, is not supported; for example, <configure><router><interface/> is not supported. An alternative is to request the parent containment node that contains the desired list node; for example, <configure><router> instead of <configure><router><interface/>.
- Content match nodes (section 6.2.5 of RFC 6241) are only supported for key leafs; for example, <configure><router><interface><interface-name>abc</interface-name>.
  - → Content match nodes that are leafs but are not also keys will result in an error (not silently ignored).

Example 1 — The following request will return an error:

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc message-id="101"
  xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
        <get-config>
                <source>
                        <running/>
                </source>
                <filter>
                        <configure>
                                <router>
                                        <interface>
                                                <interface-name>abc</interface-name>
                                                <delayed-enable>30</delayed-enable>
                                        </interface>
                                </router>
                        </configure>
                </filter>
        </get-config>
</rpc>
]]>]]>
```

Reply:

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <rpc-error>
        <error-type>application</error-type>
        <error-tag>operation-failed</error-tag>
        <error-severity>error</error-severity>
        <error-info>
            <err-element>get-config</err-element>
        </error-info>
        <error-message>
            command failed - 'configure router interface "abc" delayed-enable'
        </error-message>
    </rpc-error>
</rpc-reply>
]]>]]>
```

Multiple key leafs for the same key cannot be requested inside the same instance of the list name node; for example, <interface-name>abc</interface-name> <interface-name>def</interface-name>. Each key value must be inside its own instance of the list name node; for example, <interface> <interface-name>abc</interface-name> </interface> <interface> <interface-name>def</interface-name> </interface>.

Example 2 — A valid <get-config> request (content match on a list key):

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc message-id="101"
  xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
        <get-config>
                <source>
                        <running/>
                </source>
        <filter>
          <configure>
               <router>
               <interface>
                    <interface-name>abc</interface-name>
               </interface>
               </router>
          </configure>
        </filter>
        </get-config>
</rpc>
]]>]]>
```

Example 3 — A valid <get-config> request (selection node that is a container):

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc message-id="101"
  xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
     <get-config>
          <source>
               <running/>
          </source>
          <filter>
               <configure>
                    <router/>
```

```
            </configure>
         </filter>
     </get-config>
</rpc>
]]>]]>
```

The reply will contain all the configuration for all child nodes of config>router

Example 4 — An invalid <get-config> request (list name node - invalid selection node):

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc message-id="101"
  xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
        <get-config>
                <source>
                        <running/>
                </source>
                        <filter>
                                <configure>
                                        <router>
                                                <interface>
                                                </interface>
                                        </router>
                                </configure>
                        </filter>
        </get-config>
</rpc>
]]>]]>
```

Reply:

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <rpc-error>
        <error-type>application</error-type>
        <error-tag>operation-failed</error-tag>
        <error-severity>error</error-severity>
        <error-info>
            <err-element>get-config</err-element>
        </error-info>
        <error-message>
            command failed - 'configure router interface'
        </error-message>
    </rpc-error>
</rpc-reply>
]]>]]>
```

Example 5 — An invalid <get-config> request (empty leaf node - invalid selection node):

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc message-id="101"
  xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
        <get-config>
            <source>
                <running/>
            </source>
            <filter>
```

```
                <configure>
                <system>
                        <security>
                                    <ftp-server>
                                              </ftp-server>
                                    </security>
                         </system>
                 </configure>
            </filter>
        </get-config>
</rpc>
]]>]]>
```

Reply:

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <rpc-error>
        <error-type>protocol</error-type>
        <error-tag>bad-element</error-tag>
        <error-severity>error</error-severity>
        <error-info>
            <bad-element>ftp-server</bad-element>
        </error-info>
        <error-message>
            Element is not valid in the specified context.
        </error-message>
    </rpc-error>
</rpc-reply>
]]>]]>
```

Example 6 — An invalid <get-config> request (key repeated in the same instance of the list node):

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc message-id="101"
  xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
        <get-config>
                <source>
                        <running/>
                </source>
                <filter>
                        <configure>
                                <router>
                                        <interface>
                                                <interface-name>abc</interface-name>
                                                <interface-name>def</interface-name>
                                        </interface>
                                </router>
                        </configure>
                </filter>
        </get-config>
</rpc>
]]>]]>
```

Reply:

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <rpc-error>
        <error-type>application</error-type>
        <error-tag>operation-failed</error-tag>
        <error-severity>error</error-severity>
        <error-info>
            <err-element>get-config</err-element>
        </error-info>
        <error-message>
            command failed - 'configure router interface "abc" "def"'
        </error-message>
    </rpc-error>
</rpc-reply>
]]>]]>
```

The full configuration (equivalent to the CLI command 'admin display-config') can be obtained via a <get-config> request:

- A — when the <filter> tag is not present

  Example:

  ```
  <get-config>
       <source>
            <running/>
       </source>
  </get-config>
  ```

- B — when only the <configure> tag is present inside a <filter> tag

  Example:

  ```
  <get-config>
       <source>
            <running/>
       </source>
       <filter>
            <configure/>
       </filter>
  </get-config>
  ```

<get-config> requests that specify a non-existent list node or presence container will result in a reply that contains no data for those list nodes or containers. An rpc-error is not sent in this case.

XML Content Layer Examples

The following examples can be used after a NETCONF session has been established including the exchange of the <hello> messages.

Below is an example of a <get-config> request and response to check on whether netconf is shut down or not on the router:

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
        <get-config>
                <source> <running/> </source>
                <filter>
                        <configure>
                                <system>
                                        <netconf>
                                        </netconf>
                                </system>
                        </configure>
                </filter>
        </get-config>
</rpc>
]]>]]>
```

Reply:

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <data>
        <configure xmlns="urn:alcatel-lucent.com:sros:ns:yang:conf-r13">
            <system>
                <netconf>
                    <shutdown>false</shutdown>
                </netconf>
            </system>
        </configure>
    </data>
</rpc-reply>
]]>]]>
```

Below is an example of a <edit-config> request and response to create a basic VPRN service:

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc message-id="101"
  xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <edit-config>
        <target>
            <running/>
        </target>
        <config>
            <configure>
                <service>
                    <vprn operation="create">
                        <service-id>200</service-id>
                        <customer>1</customer>
                    </vprn>
                </service>
            </configure>
        </config>
  </edit-config>
</rpc>
]]>]]>
```

Reply:

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <ok/>
</rpc-reply>
]]>]]>
```

Below is an example of a <edit-config> request and response to create a basic VPRN service with a SAP (creates the service/interface but fails to create the SAP as the specified port's encapsulation is invalid):

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc message-id="101"
  xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <edit-config>
                <target>
            <running/>
        </target>
        <config>
            <configure>
                <service>
                    <vprn operation="create">
                        <interface>
                            <ip-int-name>"test"</ip-int-name>
                            <sap>
                                <sap-id>"2/1/1"</sap-id>
                            </sap>
                        </interface>
                        <service-id>201</service-id>
                        <customer>1</customer>
                    </vprn>
                </service>
            </configure>
        </config>
  </edit-config>
</rpc>
]]>]]>
```

Reply:

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <rpc-error>
        <error-type>application</error-type>
        <error-tag>operation-failed</error-tag>
        <error-severity>error</error-severity>
        <error-info>
            <err-element>edit-config</err-element>
        </error-info>
        <error-message>
           command failed - 'configure service vprn "201" customer 1 interface "test"
sap "2/1/1"'
            MINOR: CLI SAP-id has an invalid port number or encapsulation value.
        </error-message>
    </rpc-error>
</rpc-reply>
]]>]]>
```

CLI Content Layer

When using the CLI format at the NETCONF content layer, configuration changes and configuration information retrieved are expressed as untagged (non-XML) CLI commands; for example, CLI script.

The script must be correctly ordered and has the same dependencies and behavior as CLI. The location of CR/LF (ENTER) within the CLI for an <edit-config> is significant and affects the processing of the CLI commands, such as what CLI branch is considered the "working context". In the following two examples the "working context" after the commands are issued are different.

Example 1:

```
exit all  [<-ENTER]
configure system time zone EST [<-ENTER]
```

Example 2:

```
exit all [<-ENTER]
configure  [<-ENTER]
     system  [<-ENTER]
          time  [<-ENTER]
               zone EST [<-ENTER]
```

After example 1, the CLI working context is the root and immediately sending 'dst-zone CEST' would return an error. After example 2, the CLI working context is config>system>time and sending 'dst-zone CEST' would work as expected.

Configuration changes done via NETCONF trigger the same "change" log events (for example, tmnxConfigCreate) as a normal CLI user doing the same changes.

The <with-defaults> tag (RFC 6243) is not supported in a CLI content layer request.

The operator can get a full configuration including defaults for a CLI Content Layer using an empty <cli-info-detail>. The full configuration (equivalent to the CLI command 'admin display-config [detail]') can be obtained via a <get-config> request in a CLI Content Layer format with an empty <cli-info> or <cli-info-detail> tag inside a <config-format-cli-block>. <report-all> is not supported.

Post-processing commands are ignored: "| match" (pipe match), "| count" (pipe count) and ">" (redirect to file) and CLI ranges are not supported for any command; for example, show card [1..5].

For more information, see "CLI Content Layer Examples".

CLI Content Layer Examples

The following examples can be used after a NETCONF session has been established including the exchange of the <hello> messages.

Below is an example of a config change request and response. Note that 'exit all' at the beginning of the CLI block is not required (it is automatically assumed by the SR OS NETCONF server).

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc message-id="104" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
     <edit-config>
     <target><running/></target>
          <config>
               <config-format-cli-block>
                    configure system
                         time zone EST
                         location over-here
                    exit all
               </config-format-cli-block>
          </config>
     </edit-config>
</rpc>
]]>]]>
```

Reply:

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply message-id="104"
xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
     <ok/>
</rpc-reply>
]]>]]>
```

Below is an example of a <get-config> request and response to retrieve configuration information:

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
     <get-config>
          <source>
               <running/>
          </source>
          <filter>
               <config-format-cli-block>
                    <cli-info>router</cli-info>
                    <cli-info-detail>system login-control</cli-info-detail>
               </config-format-cli-block>
          </filter>
     </get-config>
</rpc>
]]>]]>
```

Reply:

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <data xmlns="urn:alcatel-lucent.com:sros:ns:yang:cli-content-layer-r13">
        <config-format-cli-block>
            <item>
                <cli-info>router</cli-info>
                <response>
-----------------------------------------------
#-----------------------------------------------
echo "IP Configuration"
#-----------------------------------------------
        interface "system"
            no shutdown
        exit
-----------------------------------------------
                </response>
            </item>
            <item>
                <cli-info-detail>system login-control</cli-info-detail>
                <response>
-----------------------------------------------
            ftp
                inbound-max-sessions 3
            exit
            ssh
                no disable-graceful-shutdown
                inbound-max-sessions 5
                outbound-max-sessions 5
                no ttl-security
            exit
            telnet
                no enable-graceful-shutdown
                inbound-max-sessions 5
                outbound-max-sessions 5
                no ttl-security
            exit
            idle-timeout 30
            no pre-login-message
            no motd
            login-banner
            no exponential-backoff
-----------------------------------------------
                </response>
            </item>
        </config-format-cli-block>
    </data>
</rpc-reply>
]]>]]>
```

Below is an example of a <get-config> request and response to retrieve full configuration information. Note that <cli-info-detail/> can be used to get the full configuration including default settings.

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
   <get-config>
          <source>
               <running/>
          </source>
          <filter>
              <config-format-cli-block>
                  <cli-info/>
              </config-format-cli-block>
          </filter>
    </get-config>
</rpc>
]]>]]>
```

Reply:

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <data xmlns="urn:alcatel-lucent.com:sros:ns:yang:cli-content-layer-r13">
        <config-format-cli-block>
            <item>
                <cli-info></cli-info>
                <response>
# TiMOS-C-0.0.I4301 cpm/x86_64 ALCATEL SR 7750 Copyright (c) 2000-2015 Alcatel-Lucent.
# All rights reserved. All use subject to applicable license agreements.
# Built on Sun Jan 4 19:11:11 PST 2015 by builder in /rel0.0/I4301/panos/main

# Generated WED JAN 07 01:07:43 2015 UTC

exit all
configure
#--------------------------------------------------
echo "System Configuration"
#--------------------------------------------------
    system
        chassis-mode d
        dns
        exit
        load-balancing
            lsr-load-balancing lbl-ip
            system-ip-load-balancing
        exit
        netconf
            no shutdown
        exit
        snmp
            shutdown
            engineID "deadbeefdeadbeef"
        exit
        time
            ntp
                authentication-key 1 key "OAwgNUlbzgI" hash2 type des
                no shutdown
            exit
            sntp
                shutdown
            exit
```

```
            zone EST
        exit
        thresholds
            rmon
            exit
        exit
#--------------------------------------------------
echo "Cron Configuration"
#--------------------------------------------------
        cron
              ...
              ...
              ...
        exit
    exit
#--------------------------------------------------
echo "System Security Configuration"
#--------------------------------------------------
     ...
     ...
     ...
#--------------------------------------------------
echo "System Time NTP Configuration"
#--------------------------------------------------
    system
        time
            ntp
            exit
        exit
    exit

exit all

# Finished WED JAN 07 01:07:43 2015 UTC
------------------------------------------------
------------------------------------------------
               </response>
            </item>
        </config-format-cli-block>
    </data>
</rpc-reply>
]]>]]>
```

Below is an example of a <get> request and the response to it:

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
     <get>
          <filter>
               <oper-data-format-cli-block>
                    <cli-show>system security ssh</cli-show>
               </oper-data-format-cli-block>
          </filter>
     </get>
</rpc>
]]>]]>
```

Reply:

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
    <data xmlns="urn:alcatel-lucent.com:sros:ns:yang:cli-content-layer-r13">
        <oper-data-format-cli-block>
            <item>
                <cli-show>system security ssh</cli-show>
                <response>

===============================================================================
SSH Server
===============================================================================
Administrative State      : Enabled
Operational State         : Up
Preserve Key              : Enabled

SSH Protocol Version 1    : Disabled

SSH Protocol Version 2    : Enabled
DSA Host Key Fingerprint  : ca:ce:37:90:49:7d:cc:68:22:b3:06:2c:11:cd:3c:8e
RSA Host Key Fingerprint  : 49:7c:21:97:42:35:83:61:06:95:cd:a8:78:4c:1e:76

-------------------------------------------------------------------------------
Connection                                Username
    Version Cipher                        ServerName  Status
-------------------------------------------------------------------------------
135.121.143.254                           admin
    2       aes128-cbc                    netconf     connected
-------------------------------------------------------------------------------
Number of SSH sessions : 1
===============================================================================
                </response>
            </item>
        </oper-data-format-cli-block>
    </data>
</rpc-reply>
]]>]]>
```