SIEMENS

RUGGEDCOM NETCONF

Reference Guide

05/2016
RC1065-EN-02

Disclaimer Of Liability

Siemens has verified the contents of this document against the hardware and/or software described. However, deviations between the product and the documentation may exist.

Siemens shall not be liable for any errors or omissions contained herein or for consequential damages in connection with the furnishing, performance, or use of this material.

The information given in this document is reviewed regularly and any necessary corrections will be included in subsequent editions. We appreciate any suggested improvements. We reserve the right to make technical improvements without notice.

Registered Trademarks

RUGGEDCOM™ and ROS™ are trademarks of Siemens Canada Ltd..

Linux® is the registered trademark of Linus Torvalds in the United States and other countries.

The registered trademark Linux® is used pursuant to a sublicense from LMI, the exclusive licensee of Linus Torvalds, owner of the mark on a world-wide basis.

Other designations in this manual might be trademarks whose use by third parties for their own purposes would infringe the rights of the owner.

Open Source

RUGGEDCOM ROXII is based on Linux®. Linux® is made available under the terms of the GNU General Public License Version 2.0 [http://www.gnu.org/licenses/gpl-2.0.html].

RUGGEDCOM NETCONF contains additional Open Source Software. For license conditions, refer to the associated *License Conditions* document.

Security Information

Siemens provides products and solutions with industrial security functions that support the secure operation of plants, machines, equipment and/or networks. They are important components in a holistic industrial security concept. With this in mind, Siemens' products and solutions undergo continuous development. Siemens recommends strongly that you regularly check for product updates.

For the secure operation of Siemens products and solutions, it is necessary to take suitable preventive action (e.g. cell protection concept) and integrate each component into a holistic, state-of-the-art industrial security concept. Third-party products that may be in use should also be considered. For more information about industrial security, visit http://www.siemens.com/industrialsecurity.

To stay informed about product updates as they occur, sign up for a product-specific newsletter. For more information, visit http://support.automation.siemens.com.

Contacting Siemens

Address
Siemens Canada Ltd.
Industry Sector
300 Applewood Crescent
Concord, Ontario
Canada, L4K 5C7

Telephone
Toll-free: 1 888 264 0006
Tel: +1 905 856 5288
Fax: +1 905 856 1995

E-mail
ruggedcom.info.i-ia@siemens.com

Web
www.siemens.com/ruggedcom

Table of Contents

Preface

This guide describes how to use NETCONF – the **Net**work **Conf**iguration Protocol – to manipulate configuration data on a device running the ROXII operating system.

This guide is intended to provide instructions and general guidelines for using NETCONF with ROXII. This guide is *not* intended to be a detailed guide to NETCONF itself. Where this guide introduces NETCONF concepts with which you need to be familiar, it provides links to the IETF Request For Comments (RFC) documents where the NETCONF concept or feature is discussed in detail.

Who Should Use This Guide

This guide is for network administrators and programmers tasked with the configuration management of network devices.

Readers should be familiar with the following:

- general use and function of the ROXII software.
- network design and network management concepts and tasks.
- using Secure Shell (SSH) to connect to ROXII.
- how to create well-formed and valid XML documents.

Supported Platforms

This guide describes the use of NETCONF with devices running ROXII v2.2 or higher.

Supported IETF RFCs

ROXII supports the following IETF Request For Comments (RFC):

- Internet Engineering Task Force RFC 5277 [http://tools.ietf.org/html/rfc5277]
- Internet Engineering Task Force RFC 5717 [http://tools.ietf.org/html/rfc5217]
- Internet Engineering Task Force RFC 6021 [http://tools.ietf.org/html/rfc6021]
- Internet Engineering Task Force RFC 6022 [http://tools.ietf.org/html/rfc6022]
- Internet Engineering Task Force RFC 6241 [http://tools.ietf.org/html/rfc6241]
- Internet Engineering Task Force RFC 6243 [http://tools.ietf.org/html/rfc6243]

How This Guide Is Organized

- Chapter 1, *Introducing NETCONF* introduces NETCONF and shows you what a typical NETCONF session with ROXII looks like. Read this section for a quick introduction to NETCONF on ROXII.
- Chapter 2, *NETCONF Capabilities and Namespaces* describes the NETCONF functions and data models supported by ROXII. Read this section to learn about the NETCONF functions supported by ROXII.
- Chapter 3, *NETCONF Sessions* describes how to connect to and communicate with a device with NETCONF. Read this section to learn about connecting to your device, responding to the device's initial NETCONF message, locking and unlocking datastores, and signing off from the device.
- Chapter 4, *Getting Data* describes how to retrieve configuration data from ROXII. Read this section to learn how to retrieve individual configuration elements, subsections of configuration data, or the entire configuration from the device.
- Chapter 5, *Changing Configuration Data* describes how to change ROXII configuration data. Read this section to learn how to set configuration data and perform actions.
- Chapter 6, *ROXII Actions* describes how to activate ROXII actions on a device. Read this section to learn how to activate ROXII commands, such as rebooting and clearing statistics, on the device.
- Chapter 7, *NETCONF Settings, Logs, and Statistics* describes how to configure NETCONF parameters on a device. Read this section to learn how to set NETCONF session parameters, enable NETCONF logs, and view NETCONF session statistics.
- Chapter 8, *Examples* describes many examples of how to configure ROXII data. Read this section to learn how to perform common network configuration tasks through NETCONF.
- Chapter 9, *NETCONF XML Elements* describes the XML elements unique to NETCONF commands. Read this section to learn about the XML elements used to build NETCONF commands and for information on what the elements mean when they are returned in a message from the server.

1 Introducing NETCONF

Section 1.1

What is NETCONF?

The Network Configuration Protocol (NETCONF) is a network configuration protocol developed by the Internet Engineering Task Force (IETF). NETCONF provides functions to download, upload, change, and delete the configuration data on network devices. Devices running the ROXII operating system also support the ability to collect data and perform direct actions on the device, such as rebooting the device, clearing statistics, and restarting services.

NETCONF actions and data are described using Extensible Markup Language (XML). NETCONF uses a collection of XML elements to identify functions and operations. Configuration data is represented as a hierarchy of XML elements that describe the path to a configurable setting and its data.

The NETCONF protocol can be thought of as having four conceptual layers:

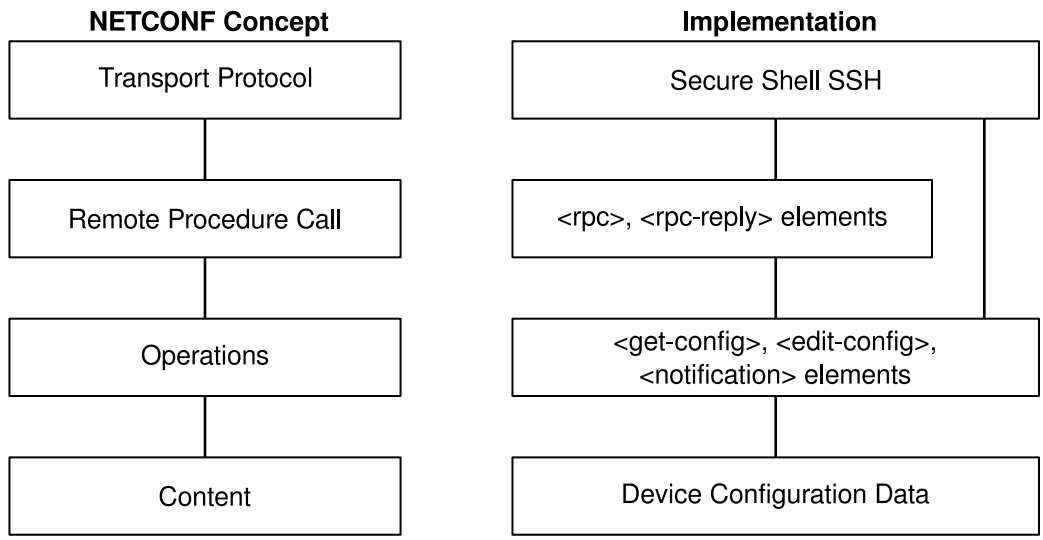

Figure 1: NETCONF Concepts and Implementation

- The *Transport Protocol* layer provides connectivity between the device and the NETCONF client. ROXII supports the use of Secure Shell (SSH) for the connection.
- The *Remote Procedure Call* layer represents NETCONF requests and responses. Requests from the client to the device are wrapped within `<rpc>` elements in the XML query text. Responses from the device to the client are wrapped within `<rpc-reply>` elements in the XML response text.
- The *Operations* layer represents actions and functions performed on the ROXII server. The operations available for use are defined by the NETCONF *capabilities* advertised by the device.
- The *Content* layer represents the configuration data on the device. NETCONF can query, manipulate, and monitor the configuration data on the device. The configuration data is defined by the RUGGEDCOM

namespaces. The configuration data is structured in NETCONF in the same way as it is in the ROXII web interface and command line interface (CLI).

The NETCONF protocol is defined in several Internet Engineering Task Force Request For Comment (RFC) documents. It is not necessary to read the RFCs to use NETCONF with devices, but this guide provides links to the RFCs for those interested in the design details of the NETCONF protocol.

For more general background information on NETCONF, refer to the Internet Engineering Task Force RFC 6241 [http://tools.ietf.org/html/rfc6241]. This RFC, published in June 2011, is the current main defining document for the NETCONF protocol.

For historical interest, refer to Internet Engineering Task Force RFC 4741 [http://tools.ietf.org/html/rfc4741]. This RFC, published in 2006, contains the initial definition of the NETCONF protocol. Note that RFC 6241 obsoletes RFC 4741.

Several additional RFCs define the NETCONF *capabilities* and *namespaces*. Links to these documents appear throughout Chapter 2, *NETCONF Capabilities and Namespaces*, where this guide discusses the capabilities and namespaces supported by devices.

Section 1.2
What Can You Do With NETCONF?

NETCONF provides an easy-to-use application programming interface (API) for ROXII. NETCONF provides the ability to view and manipulate configuration data, monitor device status, and perform device management commands.

Use NETCONF to develop custom configuration management tools and applications, such as:

- shell scripts for common device management tasks
- custom device management interfaces
- custom configuration management applications and databases
- integrating devices into existing configuration management applications and databases

Section 1.3
Sample ROXII NETCONF Sessions

This section provides a walk-through of three typical types of NETCONF sessions:

- Section 1.3.1, “Sample Session: Getting Data” describes a simple session where you connect to a device, get data from the device, and close the session.
- Section 1.3.2, “Sample Session: Performing an Action” describes a simple session where you connect to a device, perform an action on the device, and close the session.
- Section 1.3.3, “Sample Session: Editing Data” describes a more complex session where you connect to a device, prepare the device datastores for editing, edit the data, commit the data, and close the session.

Each sample provides an overview of the primary steps in the session, a schematic illustration of the primary steps, and the actual NETCONF code sent to and received from the device.

Read these sections to become familiar with the general flow of typical NETCONF sessions. Also, review these sections to become familiar with examples of working NETCONF XML code. The text in these examples can be copied and tested on an operating device.

The XML code in these examples has been formatted for legibility. Line breaks and white space have been added to the XML text to make the lines easier to read and to show the element hierarchy. When sending XML text to the device, the line breaks and whitespace are not required. You can send XML text to the device in a single line, as long as the XML is well-formed.

Text returned from the device has also been formatted for legibility. The text returned from the device typically appears in a single lines, without whitespace between the elements.

In these examples, the `<hello>` message from the device has been truncated for clarity.

Section 1.3.1
Sample Session: Getting Data

To retrieve data from a device, follow these three main steps:

1. Connect to the device and exchange `<hello>` messages.
2. Issue `<get>` or `<get-config>` commands to retrieve data from the device. You determine the data to retrieve by stating the RUGGEDCOM namespace from which you want to retrieve the data, and then stating the path through the data model to the information you want to return.
3. Close the session. Closing the session ensures that the NETCONF session closes gracefully without incomplete processes or locked datastores.

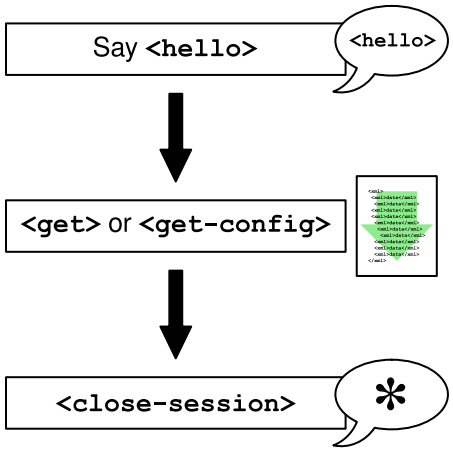

Figure 2: Session Schematic: Getting Data

1. Log in to the device via ssh:

```
$ ssh {user}@{ipAddress} -p 830 -s netconf
```

   - `{user}` is a user name on the device. Typically, the user should be assigned the administrative user role.
   - `{ipAddress}` is an address on the device listening for NETCONF activity. The `-p` parameter indicates the port listening for NETCONF activity. Port 830 is the default NETCONF port. The `-s` parameter indicates the subsystem. All NETCONF communication must be identified with `-s NETCONF`. You can configure the IP addresses and ports on which ROXII listens for NETCONF. For more information, refer to Section 7.1, “NETCONF Settings”.

   The device responds with its `<hello>` statement:

```
<?xml version="1.0" encoding="UTF-8"?>
 <hello xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <capabilities>
  .
  .
  .
  </capabilities>
  <session-id>797</session-id>
 </hello>]]>]]>
```

2. Respond to the device with the client's `<hello>` statement. The client's `<hello>` statement can describe the client's capabilities, or it can respond with just the base NETCONF capability. This example shows the minimal `<hello>` response:

```
<hello xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <capabilities>
  <capability>urn:ietf:params:netconf:base:1.0</capability>
 </capabilities>
</hello>]]>]]>
```

3. Issue an `<rpc>` request to retrieve data from the device:

```
<rpc message-id="1001" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <get-config>
  <source>
   <running/>
  </source>
  <filter type="subtree">
   <admin xmlns="http://ruggedcom.com/ns/rmf_admin">
    <system-name></system-name>
   </admin>
  </filter>
 </get-config>
</rpc>]]>]]>
```

- All commands must be enclosed within `<rpc>` tags. The `message-id` attribute is not required but is recommended. The `message-id` attribute is returned in the device response, allowing you to match responses with requests.
- The `<source>` element indicates the datastore from which we are requesting data. In this example we are requesting data from the running configuration database.
- The `<filter>` element encloses the data model tags. The `type="subtree"` attribute is mandatory.
- The `<admin>` element is the root of the RUGGEDCOM admin namespace. Within the admin element, additional elements "navigate" down to the desired element. In this example, we are navigating to `admin/system-name` in the ROXII data model.
- The `]]>]]>` string indicates the end of the NETCONF message. Each NETCONF message must end with `]]>]]>`

The device responds with the requested data:

```
<?xml version="1.0" encoding="UTF-8"?>
 <rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1001">
  <data>
   <admin xmlns="http://ruggedcom.com/ns/rmf_admin">
    <system-name>Substation Ethernet Switch 2</system-name>
   </admin>
  </data>
 </rpc-reply>]]>]]>
```

- The `<rpc-reply>` element contains the response. Notice the `message-id` attribute returned with the `<rpc-reply>` element; it corresponds to the <message-id> sent in the `<rpc>` request.

4. After receiving the data, close the session:

```
<rpc message-id="1002" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <close-session/>
</rpc>]]>]]>
```

The device responds with the following and closes the session:

```
<?xml version="1.0" encoding="UTF-8"?>
 <rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1002">
  <ok/>
 </rpc-reply>]]>]]
```

Section 1.3.2

Sample Session: Performing an Action

To perform an action on a device, such as setting the system clock, follow these three main steps:

1. Connect to the device and exchange `<hello>` messages.
2. Issue an `<rpc>` command with the action to perform. The `<rpc>` request must contain the `<action>` element referring to the action namespace. You determine the action to perform by stating the RUGGEDCOM namespace where the command is found, and then stating the path through the data model to the command.
3. Close the session. Closing the session ensures that the NETCONF session closes gracefully without incomplete processes or locked datastores.

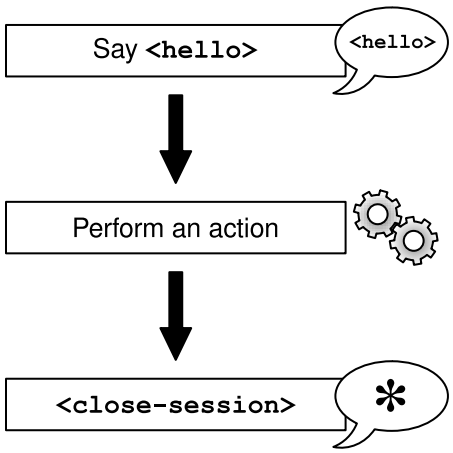

Figure 3: Session Schematic: Performing an Action

1. Log in to the device via ssh:

```
$ ssh {user}@{ipAddress} -p 830 -s netconf
```

- `{user}` is a user name on the device. Typically, the user should be assigned the administrative user role.
- `{ipAddress}` is an address on the device listening for NETCONF activity. The `-p` parameter indicates the port listening for NETCONF activity. Port 830 is the default NETCONF port. The `-s` parameter indicates the subsystem. All NETCONF communication must be identified with `-s NETCONF`. You can configure the IP addresses and ports on which ROXII listens for NETCONF. For more information, refer to Section 7.1, “NETCONF Settings”.

The device responds with its <hello> statement:

```
<?xml version="1.0" encoding="UTF-8"?>
 <hello xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <capabilities>
  .
  .
  .
  </capabilities>
  <session-id>797</session-id>
 </hello>]]>]]>
```

2. Respond to the device with the client's <hello> statement. The client's <hello> statement can describe the client's capabilities, or it can respond with just the base NETCONF capability. This example shows the minimal <hello> response:

```
<hello xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <capabilities>
  <capability>urn:ietf:params:netconf:base:1.0</capability>
 </capabilities>
</hello>]]>]]>
```

3. Issue an <rpc> request with the action to perform:

```
<rpc message-id="1005" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <action xmlns="http://tail-f.com/ns/netconf/actions/1.0">
  <data>
   <admin xmlns="http://ruggedcom.com/ns/rmf_admin">
    <set-system-clock>
     <time>2012-03-26 18:00:00</time>
    </set-system-clock>
   </admin>
  </data>
 </action>
</rpc>]]>]]>
```

- All commands must be enclosed within <rpc> tags. The message-id attribute is not required but is recommended. The message-id attribute is returned in the device response, allowing you to match responses with requests.
- The <action> element indicates that this request is to perform an action on the device. The <action> element must refer to the action namespace in the xmlns attribute.
- The <admin> element is the root of the RUGGEDCOM admin namespace. Within the admin element, additional elements "navigate" down to the desired command. In this example, we are navigating to admin/set-system-clock in the ROXII data model.
- The]]>]]> string indicates the end of the NETCONF message. Each NETCONF message must end with]]>]]>

The device responds with the results of the command:

```
<?xml version="1.0" encoding="UTF-8"?>
 <rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1005">
  <data>
   <admin xmlns='http://ruggedcom.com/ns/rmf_admin'>
    <set-system-clock>
     <system-clock>Mon Mar 26 18:00:01 2012</system-clock>
    </set-system-clock>
   </admin>
  </data>
 </rpc-reply>]]>]]>
```

- The <rpc-reply> element contains the response. Notice the message-id attribute returned with the <rpc-reply> element; it corresponds to the <message-id> sent in the <rpc> request.

4. After receiving the response, close the session:

```
<rpc message-id="1006" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <close-session/>
</rpc>]]>]]>
```

The device responds with the following and closes the session:

```
<?xml version="1.0" encoding="UTF-8"?>
 <rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1006">
  <ok/>
 </rpc-reply>]]>]]
```

Section 1.3.3

Sample Session: Editing Data

To edit data on the device, follow these main steps:

1. Connect to the device and exchange <hello> messages.
2. Issue an <rpc> command to discard changes. Discarding changes removes changes that are incomplete and not yet committed to the datastores. It is strongly recommended that you discard any such stray changes before making changes to the device configuration. Discarding changes helps to ensure that you are making changes to a known state of the configuration.
3. Issue an <rpc> command to lock the candidate and running datastores. Locking the datastores prevents other users in other sessions from editing the database while the NETCONF session is working with it. It is strongly recommended that you lock the datastores before making changes to the device configuration.
4. Issue an <rpc> command to edit the configuration. You determine which data to edit by stating the RUGGEDCOM namespace where the data to be changed is found, and then stating the path through the data model to the items to change.
5. Issue an <rpc> command to validate the changes. Validating the changes ensures that the syntax of the changes is correct.
6. Issue an <rpc> command to commit the changes. Committing the changes applies the changes to the running configuration, making the changes take effect on the running device.
7. Issue an <rpc> command to unlock the datastores. Unlock the datastores to allow other users in other sessions to modify the configuration data.
8. Close the session. Closing the session ensures that the NETCONF session closes gracefully without incomplete processes or locked datastores.

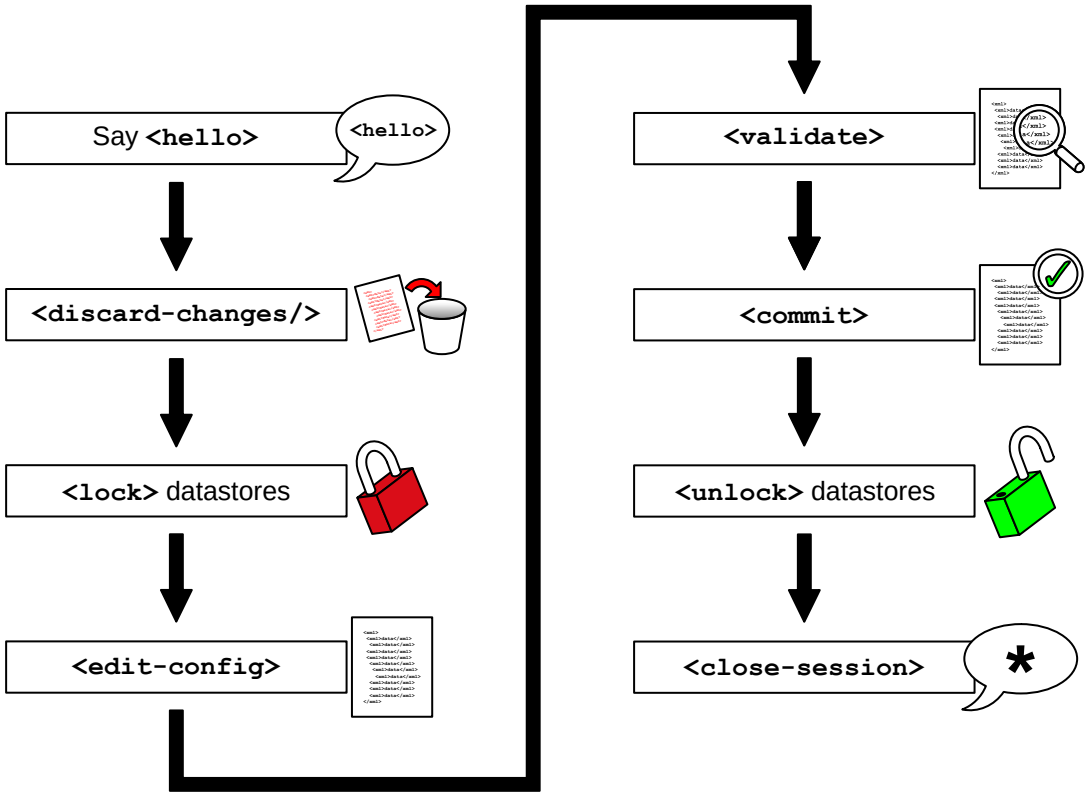

Figure 4: Session Schematic: Editing Data

The following procedure provides more details:

1. Log in to the device via ssh:

```
$ ssh {user}@{ipAddress} -p 830 -s netconf
```

   - `{user}` is a user name on the device. Typically, the user should be assigned the administrative user role.
   - `{ipAddress}` is an address on the device listening for NETCONF activity. The `-p` parameter indicates the port listening for NETCONF activity. Port 830 is the default NETCONF port. The `-s` parameter indicates the subsystem. All NETCONF communication must be identified with `-s NETCONF`. You can configure the IP addresses and ports on which ROXII listens for NETCONF. For more information, refer to Section 7.1, “NETCONF Settings”.

   The device responds with its `<hello>` statement:

```
<?xml version="1.0" encoding="UTF-8"?>
 <hello xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <capabilities>
  .
  .
  .
  </capabilities>
  <session-id>797</session-id>
 </hello>]]>]]>
```

2. Respond to the device with the client’s `<hello>` statement. The client’s `<hello>` statement can describe the client’s capabilities, or it can respond with just the base NETCONF capability. This example shows the minimal `<hello>` response:

```
<hello xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <capabilities>
  <capability>urn:ietf:params:netconf:base:1.0</capability>
 </capabilities>
</hello>]]>]]>
```

3. Issue an <rpc> request to discard configuration changes:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1012">
 <discard-changes/>
</rpc>
]]>]]>
```

   - The <discard-changes/> command discards any uncommitted changes that may be present in the configuration. It is recommended that you perform this step to ensure that the changes you make are made to a known state of the configuration.

   The device responds with the following:

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1012">
 <ok/>
</rpc-reply>]]>]]>
```

   Any uncommitted changes are removed from the configuration.

4. Issue an <rpc> request to lock the running configuration:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1010">
 <lock>
  <target>
   <running/>
  </target>
 </lock>
</rpc>
]]>]]>
```

   - All commands must be enclosed within <rpc> tags. The message-id attribute is not required but is recommended. The message-id attribute is returned in the device response, allowing you to match responses with requests.
   - The <lock> element indicates that this request is to lock a configuration.
   - The <target> element specifies the configuration to lock. In this <rpc>, the lock target is the <running/> configuration.
   - The]]>]]> string indicates the end of the NETCONF message. Each NETCONF message must end with]]>]]>

   The device responds with the following:

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1010">
 <ok/>
</rpc-reply>]]>]]>
```

   The running configuration is now locked.

5. Issue an <rpc> request to lock the candidate configuration:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1011">
 <lock>
  <target>
   <candidate/>
  </target>
```

```
 </lock>
</rpc>]]>]]>
```

The device responds with the following:

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1011">
 <ok/>
</rpc-reply>]]>]]>
```

The candidate configuration is now locked.

6. Issue an <rpc> request to edit the candidate configuration:

```
<rpc message-id="1014" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <edit-config>
  <target>
   <candidate/>
  </target>
  <config>
   <admin xmlns="http://ruggedcom.com/ns/rmf_admin">
    <system-name>Substation Ethernet: Switch 01</system-name>
   </admin>
  </config>
 </edit-config>
</rpc>]]>]]>
```

- The <edit-config> element indicates that this request is to edit the configuration.
- The <target> element specifies the configuration to be edited. In this example, the <candidate/> configuration is to be edited.
- The <config> element contains the path to the value to be edited.
- The <admin> element specifies that the value to be edited is in the RUGGEDCOM admin namespace. In this example, the <system-name> value is being edited.

The device responds with the following:

```
<?xml version="1.0" encoding="UTF-8"?>
 <rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1014">
  <ok/>
 </rpc-reply>]]>]]>
```

The edit is applied to the <candidate/> configuration.

7. Issue an <rpc> request to validate the candidate configuration:

```
<rpc message-id="1015" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <validate>
  <source>
   <candidate/>
  </source>
 </validate>
</rpc>]]>]]>
```

- The <validate> element indicates that this request is to validate a specified configuration.
- The <source> element specifies the configuration to be validated. In this example, the <candidate/> configuration is to be validated.

The device responds with the following:

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1015">
 <ok/>
</rpc-reply>]]>]]>
```

The candidate configuration is validated and its syntax is found to be correct. Had there been syntax errors, the device would return a message with <error-type>, <error-tag>, <error-severity>, <error-path>, <error-info>, and <bad-element> elements to describe and identify the syntax error.

8. Issue an <rpc> request to commit the changes:

```
<rpc message-id="1016" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <commit/>
</rpc>]]>]]>
```

The device responds with the following:

```
<?xml version="1.0" encoding="UTF-8"?>
 <rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1016">
  <ok/>
 </rpc-reply>]]>]]>
```

The changes made to the <candidate/> configuration are committed and promoted to the <running/> configuration.

9. Issue an <rpc> request to unlock the candidate configuration:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1017">
 <unlock>
  <target>
   <candidate/>
  </target>
 </unlock>
</rpc>]]>]]>
```

The device responds with the following:

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1017">
 <ok/>
</rpc-reply>]]>]]>
```

10. Issue an <rpc> request to unlock the running configuration:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1018">
 <unlock>
  <target>
   <running/>
  </target>
 </unlock>
</rpc>]]>]]>
```

The device responds with the following:

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1018">
 <ok/>
</rpc-reply>]]>]]>
```

11. Close the session:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1019" >
 <close-session/>
</rpc>]]>]]>
```

The device responds with the following and closes the session:

```
<?xml version="1.0" encoding="UTF-8"?>
 <rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1019">
  <ok/>
 </rpc-reply>]]>]]
```

2 NETCONF Capabilities and Namespaces

This section describes the capabilities supported by the ROXII software.

NETCONF capabilities describe the functions and namespaces supported by a NETCONF peer. When you connect to the NETCONF service on a device, the device advertises its capabilities in a `<hello>` message.

Capabilities and namespaces are reported within `<capability>` elements in the `<hello>` message. A `<capability>` element describes a capability or a namespace:

- a *capability* is a function or service provided by the device. For example, the ability to commit changes to the database or to lock a portion of the database are *capabilities*.
- a *namespace* is a definition of data elements. For example, the definition of standard Internet address elements and the definition of ROXII configuration parameters are *namespaces*.

ROXII supports both standard IETF NETCONF capabilities and vendor-defined capabilities that are unique to the product platform.

ROXII uses namespaces that define the ROXII configuration data model and that support various capabilities.

Section 2.1

IETF Capabilities

The following are the standard IETF capabilities supported by ROXII. These capabilities define most of the actions that can be performed through NETCONF on a device.

- **<capability>urn:ietf:params:netconf:base:1.0</capability>**
  This is the base NETCONF capability. When replying to the `<hello>` message from a device, the NETCONF client must respond with at least this capability.

  For more information on this capability, see Internet Engineering Task Force RFC 6241 [http://tools.ietf.org/html/rfc6241].
- **<capability>urn:ietf:params:netconf:capability:writable-running:1.0</capability>**
  Supports writing to the running configuration: you can update configuration data in the running configuration.

  For more information on this capability, see Internet Engineering Task Force RFC 6241 [http://tools.ietf.org/html/rfc6241].
- **<capability>urn:ietf:params:netconf:capability:candidate:1.0</capability>**
  Supports a candidate configuration: you can make changes to a candidate configuration, validate and review the changes, and then commit the candidate to make it the running configuration.

  For more information on this capability, see Internet Engineering Task Force RFC 6241 [http://tools.ietf.org/html/rfc6241].
- **<capability>urn:ietf:params:netconf:capability:confirmed-commit:1.0</capability>**
  Supports the confirmed commit operation: you can require that a commit be confirmed before a candidate configuration is promoted to become the running configuration.

  For more information on this capability, see Internet Engineering Task Force RFC 6241 [http://tools.ietf.org/html/rfc6241].

- **<capability>urn:ietf:params:netconf:capability:XPath:1.0</capability>**
  Supports the use of XPath expressions: you can used XPath expressions in the <filter> element to define the path to the configuration item to be retrieved or set.

  For more information on this capability, see Internet Engineering Task Force RFC 6241 [http://tools.ietf.org/html/rfc6241].
- **<capability>urn:ietf:params:netconf:capability:url:1.0?scheme=ftp,sftp,file</capability>**
  Supports file transfer for configuration data: you can upload or download configuration data as a file through a specified protocol.

  For more information on this capability, see Internet Engineering Task Force RFC 6241 [http://tools.ietf.org/html/rfc6241].
- **<capability>urn:ietf:params:netconf:capability:validate:1.0</capability>**
  Supports the validate operation: you can validate a specified configuration for syntax errors.

  For more information on this capability, see Internet Engineering Task Force RFC 6241 [http://tools.ietf.org/html/rfc6241].
- **<capability>urn:ietf:params:netconf:capability:rollback-on-error:1.0</capability>**
  Supports the rollback-on-error operation: you can require the configuration to roll back to its previous state if a commit fails.

  For more information on this capability, see Internet Engineering Task Force RFC 6241 [http://tools.ietf.org/html/rfc6241].
- **<capability>urn:ietf:params:netconf:capability:notification:1.0</capability>**
  Supports the notification operation: you can have the NETCONF server advise a NETCONF client of changes to the configuration data or device state.

  For more information on this capability, see Internet Engineering Task Force RFC 5277 [http://tools.ietf.org/html/rfc5277].
- **<capability>urn:ietf:params:netconf:capability:interleave:1.0</capability>**
  Supports the interleave capability: the device handles NETCONF notification messages and other NETCONF operations asynchronously.

  For more information on this capability, see Internet Engineering Task Force RFC 5277 [http://tools.ietf.org/html/rfc5277].
- **<capability>urn:ietf:params:netconf:capability:partial-lock:1.0</capability>**
  Supports the partial-lock capability: you can lock a specified portion of the configuration database for updating.

  For more information on this capability, see Internet Engineering Task Force RFC 5717 [http://tools.ietf.org/html/rfc5717].
- **<capability>urn:ietf:params:netconf:capability:with-defaults:1.0?basic-mode=trim&also-supported=report-all-tagged</capability>**
  Supports the with-defaults capability: you can control how the NETCONF server reports default data in the data model.

  For more information on this capability, see Internet Engineering Task Force RFC 6243 [http://tools.ietf.org/html/rfc6243].

Section 2.2
Vendor-Defined Capabilities

The following capabilities are provided by the vendor of the development tools used to create the ROXII software. These vendor-defined capabilities complement and extend the standard IETF capabilities.

- **<capability>http://tail-f.com/ns/netconf/with-defaults/1.0</capability>**
  This vendor-defined capability extends the standard IETF with-defaults capability.
- **<capability>http://tail-f.com/ns/netconf/actions/1.0</capability>**
  This vendor-defined capability supports the execution of actions on the device: you can issue direct commands through NETCONF, such as reboot, clear-all-alarms, restore-factory-defaults, and others.
- **<capability>http://tail-f.com/ns/netconf/commit/1.0</capability>**
  This vendor-defined capability extends the commit capability: you can make changes to a candidate configuration and commit the changes to promote them to the running configuration.
- **<capability>http://tail-f.com/yang/common-monitoring?module=tailf-common-monitoring&revision=2013-06-14</capability>**
  <capability>http://tail-f.com/yang/confd-monitoring?module=tailf-confd-monitoring&revision=2013-06-14</capability>
  <capability>http://tail-f.com/yang/netconf-monitoring?module=tailf-netconf-monitoring&revision=2012-06-14</capability>

  These vendor-defined capabilities support NETCONF monitoring on the device.

Section 2.3
IETF Namespaces

ROXII uses several namespaces to data types and configuration data models. Some namespaces are associated with and provide support for specific NETCONF capabilities.

The following are the standard IETF namespaces supported by ROXII:

- **<capability>urn:ietf:params:xml:ns:yang:ietf-inet-types?module=ietf-inet-types&revision=2010-09-24</capability>**
  Defines data types for Internet addresses and related items.

  For more information on this namespace, see Internet Engineering Task Force RFC 6021 [http://tools.ietf.org/html/rfc6021].
- **<capability>urn:ietf:params:xml:ns:yang:ietf-netconf-monitoring?module=ietf-netconf-monitoring&revision=2010-10-04</capability>**
  Defines data types for NETCONF monitoring.

  For more information on this namespace, see Internet Engineering Task Force RFC 6022 [http://tools.ietf.org/html/rfc6022].
- **<capability>urn:ietf:params:xml:ns:yang:ietf-yang-types?module=ietf-yang-types&revision=2010-09-24</capability>**
  Defines data types for general YANG data types. YANG is the data modelling language used to develop the ROXII software.

  For more information on this namespace, see Internet Engineering Task Force RFC 6022 [http://tools.ietf.org/html/rfc6022].
- **<capability>urn:ietf:params:xml:ns:yang:ietf-netconf-with-defaults?revision=2010-11-11&module=ietf-with-defaults</capability>**
  Defines items used by the with-defaults capability.

  For more information on this namespace, see Internet Engineering Task Force RFC 6243 [http://tools.ietf.org/html/rfc6243].

Section 2.4
Vendor-defined Namespaces

The following namespaces support vendor-defined NETCONF capabilities:

- **<capability>http://tail-f.com/ns/netconf/with-defaults/1.0</capability>**
  Supports and extends the IETF with-defaults capability.
- **<capability>http://tail-f.com/ns/netconf/actions/1.0</capability>**
  Supports the vendor-defined actions capability.
- **<capability>http://tail-f.com/ns/netconf/commit/1.0</capability>**
  Supports the vendor-defined commit capability.

Section 2.5
RUGGEDCOM Namespaces

The RUGGEDCOM namespaces define the configuration data model on the device. Depending on the physical configuration of your device, not all RUGGEDCOM namespaces may be present. For example, if your device does not have switch interfaces, the switch namespace will not be present.

- **<capability>http://ruggedcom.com/ns/rmf?module=rmf&revision=2012-11-28</capability>**
  The parent container for all ROXII configuration data.
- **<capability>http://ruggedcom.com/ns/rmf_admin?module=rmf_admin&revision=2012-11-28</capability>**
  The admin namespace contains administrative configuration data. The admin namespace is the equivalent of the **admin** menu level in the ROXII web user interface, and the **`admin`** command in the ROXII command line interface.
- **<capability>http://ruggedcom.com/ns/rmf_chassis?module=rmf_chassis&revision=2012-11-28</capability>**
  The chassis namespace contains chassis configuration data. The chassis namespace is the equivalent of the **chassis** menu level in the ROXII web user interface, and the **`chassis`** command in the ROXII command line interface.
- **<capability>http://ruggedcom.com/ns/rmf_crossbow?module=rmf_crossbow&revision=2013-10-10</capability>**
  The crossbow namespace contains CROSSBOW configuration data. The crossbow namespace is the equivalent of the **apps/crossbow** menu level in the ROXII web user interface, and the **`crossbow`** command in the ROXII command line interface.
- **<capability>http://ruggedcom.com/ns/rmf_elan?module=rmf_elan&revision=2012-11-28</capability>**
  The elan namespace contains ELAN configuration data. The elan namespace is the equivalent of the **apps/elan** menu level in the ROXII web user interface, and the **`elan`** command in the ROXII command line interface.
- **<capability>http://ruggedcom.com/ns/rmf_events?module=rmf_events&revision=2012-11-28</capability>**
  The events namespace contains event configuration data.
- **<capability>http://ruggedcom.com/ns/rmf_global?module=rmf_global&revision=2012-11-28</capability>**
  The global namespace contains global configuration data. The global namespace is the equivalent of the **global** menu level in the ROXII web user interface, and the **`global`** command in the ROXII command line interface.

- **<capability>http://ruggedcom.com/ns/rmf_if?module=rmf_if&revision=2012-11-28</capability>**
  The interface namespace contains interface configuration data. The interface namespace is the equivalent of the **interface** menu level in the ROXII web user interface, and the **`interface`** command in the ROXII command line interface.
- **<capability>http://ruggedcom.com/ns/rmf_ifs?module=rmf_ifs&revision=2012-11-28</capability>**
  The interfaces namespace contains interfaces read-only data. The interface namespace is the equivalent of the **interfaces** menu level in the ROXII web user interface, and the **`interfaces`** command in the ROXII command line interface.
- **<capability>http://ruggedcom.com/ns/rmf_ifswitch?module=rmf_ifswitch&revision=2012-11-28</capability>**
  The switch namespace contains switch configuration data. The switch namespace is the equivalent of the **switch** menu level in the ROXII web user interface, and the **`switch`** command in the ROXII command line interface.
- **<capability>http://ruggedcom.com/ns/rmf_iftunnel?module=rmf_iftunnel&revision=2012-11-28</capability>**
  The tunnel namespace contains tunnel configuration data. The tunnel namespace is the equivalent of the **tunnel** menu level in the ROXII web user interface, and the **`tunnel`** command in the ROXII command line interface.
- **<capability>http://ruggedcom.com/ns/rmf_ip?module=rmf_ip&revision=2012-11-28</capability>**
  The ip namespace contains ip address configuration data. The ip namespace is the equivalent of the **ip** menu level in the ROXII web user interface, and the **`ip`** command in the ROXII command line interface.
- **<capability>http://ruggedcom.com/ns/rmf_mpls?module=rmf_mpls&revision=2012-11-28</capability>**
  The mpls namespace contains mpls configuration data. The mpls namespace is the equivalent of the **mpls** menu level in the ROXII web user interface, and the **`mpls`** command in the ROXII command line interface.
- **<capability>http://ruggedcom.com/ns/rmf_qos?module=rmf_qos&revision=2012-11-28</capability>**
  The qos namespace contains quality of service configuration data. The qos namespace is the equivalent of the **qos** menu level in the ROXII web user interface, and the **`qos`** command in the ROXII command line interface.
- **<capability>http://ruggedcom.com/ns/rmf_routing?module=rmf_routing&revision=2012-11-28</capability>**
  The routing namespace contains routing configuration data. The routing namespace is the equivalent of the **routing** menu level in the ROXII web user interface, and the **`routing`** command in the ROXII command line interface.
- **<capability>http://ruggedcom.com/ns/rmf_security?module=rmf_security&revision=2012-11-28</capability>**
  The security namespace contains security configuration data. The security namespace is the equivalent of the **security** menu level in the ROXII web user interface, and the **`security`** command in the ROXII command line interface.
- **<capability>http://ruggedcom.com/ns/rmf_services?module=rmf_services&revision=2012-11-28</capability>**
  The services namespace contains services configuration data. The services namespace is the equivalent of the **services** menu level in the ROXII web user interface, and the **`services`** command in the ROXII command line interface.
- **<capability>http://ruggedcom.com/ns/rox_apps?module=rox_apps&revision=2012-11-28</capability>**
  The apps namespace contains apps configuration data. The apps namespace is the equivalent of the **apps** menu level in the ROXII web user interface, and the **`apps`** command in the ROXII command line interface.
- **<capability>http://ruggedcom.com/ns/rmf_mpls?module=rmf_mpls&revision=2012-11-28</capability>**
  The mpls namespace contains mpls configuration data. The mpls namespace is the equivalent of the **mpls** menu level in the ROXII web user interface, and the **`mpls`** command in the ROXII command line interface.

Section 2.6

Viewing the Capabilities on a Device

To view the capabilities on a device, do the following:

1. Log in to the device's NETCONF service via Secure Shell (SSH):

```
$ ssh {user}@{ipAddress} -p 830 -s netconf
```

   - `{user}` is an administrative role user on the device.
   - `{ipAddress}` is an address on the device listening for NETCONF activity. The `-p` parameter indicates the port listening for NETCONF activity. Port 830 is the default NETCONF port. The `-s` parameter indicates the subsystem. All NETCONF communication must be identified with `-s NETCONF`. You can configure the IP addresses and ports on which ROXII listens for NETCONF. For more information, refer to Section 7.1, "NETCONF Settings".
2. When prompted, provide the user's password.
3. The device responds with a `<hello>` message, listing its capabilities.

```
<?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</capability>
   <capability>urn:ietf:params:netconf:capability:candidate:1.0</capability>
   <capability>urn:ietf:params:netconf:capability:confirmed-commit:1.0</capability>
   <capability>urn:ietf:params:netconf:capability:confirmed-commit:1.1</capability>
   <capability>urn:ietf:params:netconf:capability:XPath:1.0</capability>
   <capability>urn:ietf:params:netconf:capability:url:1.0?scheme=ftp,sftp,file</capability>
   <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:rollback-on-error:1.0</capability>
   <capability>urn:ietf:params:netconf:capability:notification:1.0</capability>
   <capability>urn:ietf:params:netconf:capability:interleave:1.0</capability>
   <capability>urn:ietf:params:netconf:capability:partial-lock:1.0</capability>
   <capability>http://tail-f.com/ns/netconf/with-defaults/1.0</capability>
   <capability>http://tail-f.com/ns/netconf/actions/1.0</capability>
   <capability>http://tail-f.com/ns/netconf/commit/1.0</capability>
   <capability>urn:ietf:params:netconf:capability:with-defaults:1.0?basic-mode=trim&also-
supported=report-all-tagged</capability>
   <capability>urn:ietf:params:xml:ns:yang:ietf-netconf-with-defaults?
revision=2010-11-11&module=ietf-with-defaults</capability>
   <capability>http://ruggedcom.com/ns/rmf?module=rmf&revision=2012-03-07</capability>
   <capability>http://ruggedcom.com/ns/rmf_admin?module=rmf_admin&revision=2012-03-07</capability>
   <capability>http://ruggedcom.com/ns/rmf_chassis?module=rmf_chassis&revision=2012-03-07</
capability>
   <capability>http://ruggedcom.com/ns/rmf_events?module=rmf_events&revision=2012-03-07</
capability>
   <capability>http://ruggedcom.com/ns/rmf_global?module=rmf_global&revision=2012-03-07</
capability>
   <capability>http://ruggedcom.com/ns/rmf_if?module=rmf_if&revision=2012-03-07</capability>
   <capability>http://ruggedcom.com/ns/rmf_ifs?module=rmf_ifs&revision=2012-03-07</capability>
   <capability>http://ruggedcom.com/ns/rmf_iftunnel?module=rmf_iftunnel&revision=2012-03-07</
capability>
   <capability>http://ruggedcom.com/ns/rmf_ip?module=rmf_ip&revision=2012-03-07</capability>
   <capability>http://ruggedcom.com/ns/rmf_qos?module=rmf_qos&revision=2012-03-07</capability>
   <capability>http://ruggedcom.com/ns/rmf_routing?module=rmf_routing&revision=2012-03-07</
capability>
   <capability>http://ruggedcom.com/ns/rmf_security?module=rmf_security&revision=2012-03-07</
capability>
```

```
   <capability>http://ruggedcom.com/ns/rmf_services?module=rmf_services&revision=2012-03-07</
capability>
   <capability>http://tail-f.com/yang/common-monitoring?module=tailf-common-
monitoring&revision=2011-09-22</capability>
   <capability>http://tail-f.com/yang/confd-monitoring?module=tailf-confd-
monitoring&revision=2011-09-22</capability>
   <capability>http://tail-f.com/yang/netconf-monitoring?module=tailf-netconf-
monitoring&revision=2011-09-22</capability>
   <capability>urn:ietf:params:xml:ns:yang:ietf-inet-types?module=ietf-inet-
types&revision=2010-09-24</capability>
   <capability>urn:ietf:params:xml:ns:yang:ietf-netconf-monitoring?module=ietf-netconf-
monitoring&revision=2010-10-04</capability>
   <capability>urn:ietf:params:xml:ns:yang:ietf-yang-types?module=ietf-yang-
types&revision=2010-09-24</capability>
  </capabilities>
  <session-id>1020</session-id>
 </hello>]]>]]>
```

3 NETCONF Sessions

This section describes how to do the following:

- connect to the NETCONF service on a device via Secure Shell (SSH). You must do this each time you connect to the device to start a NETCONF session.
- respond to the device's `<hello>` message. You must do this each time you start a NETCONF session.
- close the session gracefully with the `<close-session>` command. Closing the session with the `<close-session>` command is recommended, but is optional.
- kill the session with the `<kill-session>` command. Closing the session with the `<kill-session>` command is optional.

Section 3.1

Connecting to the NETCONF Service

ROXII suports the use of Secure Shell (SSH) to connect to the NETCONF service on the device. To connect to the NETCONF service, specify an IP address, port, and subsystem:

```
$ ssh {user}@{ipAddress} -p 830 -s netconf
```

- `{user}`

  A user name on the device assigned the administrative user role.

- `{ipAddress}`

  The IP address or hostname of the device.

- `-p 830`

  Specifies port 830. The default NETCONF port is port 830.

- `-s netconf`

  Specifies the NETCONF subsystem. You must always specify the NETCONF subsystem when initiating a NETCONF session.

You can configure the IP addresses and ports on which ROXII listens for NETCONF. For more information, refer to Section 7.1, "NETCONF Settings".

When prompted, provide the user password.

When you connect to the device, the device responds with a `<hello>` message, listing its NETCONF capabilities. For more information on the `<hello>` message and on how to respond to it, refer to Section 3.2, "Saying Hello".

Section 3.2
Saying Hello

When you open a NETCONF session, the device responds with a <hello> message. The <capabilities> elements in the message list the NETCONF commands, functions, and namespaces supported on the device. The <hello> message also includes a <session-id> element, containing a numeric identifier for the new session.

Before you can issue NETCONF requests, you must respond to the <hello> message. The minimal response is to reply with a <hello> message listing just the **netconf:base** capability from the client. You can also reply with the client's actual capabilities, or reply by returning the device's capabilities back to the device. In all examples in this guide, we respond to the <hello> message with the minimal response.

NOTE
Your reply to the <hello> *message must not contain a* <session-id>*. Including a* <session-id> *in your response results in a bad element error and the device closes the session.*

If you choose to return the device's <hello> message as the response, make sure that only one version of each capability is present in your response. ROXII advertises two versions of the following 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:confirmed-commit:1.0</capability>
<capability>urn:ietf:params:netconf:capability:confirmed-commit:1.1</capability>

<capability>urn:ietf:params:netconf:capability:validate:1.0</capability>
<capability>urn:ietf:params:netconf:capability:validate:1.1</capability>
```

You must return either version 1.0 or version 1.1 of each capability. Do not return both versions of each capability.

Section 3.2.1
ROXII NETCONF <hello> Message

The following is the hello message returned when you connect to the NETCONF service on a device running the ROXII operating system:

```
<?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</capability>
   <capability>urn:ietf:params:netconf:capability:candidate:1.0</capability>
   <capability>urn:ietf:params:netconf:capability:confirmed-commit:1.0</capability>
   <capability>urn:ietf:params:netconf:capability:confirmed-commit:1.1</capability>
   <capability>urn:ietf:params:netconf:capability:xpath:1.0</capability>
   <capability>urn:ietf:params:netconf:capability:url:1.0?scheme=ftp,sftp,file</capability>
   <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:rollback-on-error:1.0</capability>
   <capability>urn:ietf:params:netconf:capability:notification:1.0</capability>
   <capability>urn:ietf:params:netconf:capability:interleave:1.0</capability>
   <capability>urn:ietf:params:netconf:capability:partial-lock:1.0</capability>
   <capability>http://tail-f.com/ns/netconf/with-defaults/1.0</capability>
   <capability>http://tail-f.com/ns/netconf/actions/1.0</capability>
   <capability>http://tail-f.com/ns/netconf/commit/1.0</capability>
```

```
   <capability>urn:ietf:params:netconf:capability:with-defaults:1.0?basic-mode=trim&also-
supported=report-all-tagged</capability>
   <capability>urn:ietf:params:xml:ns:yang:ietf-netconf-with-defaults?revision=2010-11-11&module=ietf-
with-defaults</capability>
   <capability>http://ruggedcom.com/ns/rmf?module=rmf&revision=2012-03-07</capability>
   <capability>http://ruggedcom.com/ns/rmf_admin?module=rmf_admin&revision=2012-03-07</capability>
   <capability>http://ruggedcom.com/ns/rmf_chassis?module=rmf_chassis&revision=2012-03-07</capability>
   <capability>http://ruggedcom.com/ns/rmf_events?module=rmf_events&revision=2012-03-07</capability>
   <capability>http://ruggedcom.com/ns/rmf_global?module=rmf_global&revision=2012-03-07</capability>
   <capability>http://ruggedcom.com/ns/rmf_if?module=rmf_if&revision=2012-03-07</capability>
   <capability>http://ruggedcom.com/ns/rmf_ifs?module=rmf_ifs&revision=2012-03-07</capability>
   <capability>http://ruggedcom.com/ns/rmf_iftunnel?module=rmf_iftunnel&revision=2012-03-07</
capability>
   <capability>http://ruggedcom.com/ns/rmf_ip?module=rmf_ip&revision=2012-03-07</capability>
   <capability>http://ruggedcom.com/ns/rmf_qos?module=rmf_qos&revision=2012-03-07</capability>
   <capability>http://ruggedcom.com/ns/rmf_routing?module=rmf_routing&revision=2012-03-07</capability>
   <capability>http://ruggedcom.com/ns/rmf_security?module=rmf_security&revision=2012-03-07</
capability>
   <capability>http://ruggedcom.com/ns/rmf_services?module=rmf_services&revision=2012-03-07</
capability>
   <capability>http://tail-f.com/yang/common-monitoring?module=tailf-common-
monitoring&revision=2011-09-22</capability>
   <capability>http://tail-f.com/yang/confd-monitoring?module=tailf-confd-
monitoring&revision=2011-09-22</capability>
   <capability>http://tail-f.com/yang/netconf-monitoring?module=tailf-netconf-
monitoring&revision=2011-09-22</capability>
   <capability>urn:ietf:params:xml:ns:yang:ietf-inet-types?module=ietf-inet-types&revision=2010-09-24</
capability>
   <capability>urn:ietf:params:xml:ns:yang:ietf-netconf-monitoring?module=ietf-netconf-
monitoring&revision=2010-10-04</capability>
   <capability>urn:ietf:params:xml:ns:yang:ietf-yang-types?module=ietf-yang-types&revision=2010-09-24</
capability>
  </capabilities>
  <session-id>1020</session-id>
 </hello>]]>]]>
```

The following is the minimal hello response required from the NETCONF client:

```
<hello xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <capabilities>
  <capability>urn:ietf:params:netconf:base:1.0</capability>
 </capabilities>
</hello>]]>]]>
```

The device does not reply to the `<hello>` response unless there is an error. After issuing the `<hello>` response, you can begin sending NETCONF requests.

Section 3.3

Closing the Session

Closing a session requests the graceful termination of a NETCONF session. It is recommended that you use the `<close-session/>` command to close each NETCONF session. Upon closing the session, the device also terminates the SSH connection.

When the server receives a `<close-session/>` request, it does the following:

- gracefully closes the session
- releases any locks and resources associated with the session
- gracefully closes any associated connections
- ignores any NETCONF requests received after the `<close-session/>` request

If the NETCONF device can complete the request, it sends an <rpc-reply> document containing the <ok/> element.

If the NETCONF device cannot complete the request, it sends an <rpc-reply> document containing the <rpc-error> element.

To close a NETCONF session, send the following:

```
<rpc message-id="2010" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <close-session/>
</rpc>]]>]]>
```

Upon successfully closing the session, the device responds with the following:

```
<?xml version="1.0" encoding="UTF-8"?>
 <rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="2010">
  <ok/>
 </rpc-reply>]]>]]>
```

Section 3.4

Killing a Session

Killing a session terminates a specified NETCONF session, cancelling any operations in progress and releasing all locks, resources, and connections for the session. Use the <kill-session> command to close NETCONF sessions *other* than the current session. You cannot use <kill-session> to close the session from which the command is used.

<kill-session> does not roll back the configuration or state changes made by the configuration being terminated. If the session being terminated is performing a confirmed commit when the <kill-session> is issued, the NETCONF server restores the configuration to its state before the confirmed commit was issued.

To kill a session, you need to know its <session-id>. To determine the <session-id> of another session, attempt to <lock> a configuration. If the configuration is already locked by another session, the <session-id> for the session is reported in the <rpc-error> message received from the unsuccessful <lock> attempt.

To kill a session where you know the <session-id>, issue the following:

```
<rpc message-id="2030" mlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <kill-session>
  <session-id>1968</session-id>
 </kill-session>
</rpc>>>>>]]>
```

The device responds with the following message and kills the session:

```
<rpc-reply message-id="2030" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <ok/>
</rpc-reply>
```

To kill a session where you do not know the <session-id>, first attempt to <lock> a configuration. In this example, we attempt to lock the already locked <running/> configuration:

```
<rpc message-id="2040" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <lock>
  <target>
   <running/>
  </target>
 </lock>
</rpc>]]>]]>
```

The device responds with the following:

```
<?xml version="1.0" encoding="UTF-8"?>
 <rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="2040">
 <rpc-error>
  <error-type>protocol</error-type>
  <error-tag>lock-denied</error-tag>
  <error-severity>error</error-severity>
  <error-info>
   <session-id>1968</session-id>
  </error-info>
 </rpc-error>
</rpc-reply>]]>]]>
```

Incorporate the `<session-id>` into the `<kill-session>` command:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="2041">
 <kill-session>
  <session-id>1968</session-id>
 </kill-session>
</rpc>
```

The device responds with the following message and kills the specified session:

```
<rpc-reply message-id="2041" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <ok/>
</rpc-reply>
```

4 Getting Data

This section describes how to use NETCONF to retrieve configuration data from ROXII.

NETCONF features two get commands to retrieve data from the device:

- `<get>` retrieves device state data and information from the running configuration. For more information, refer to Section 4.1, “Using the <get> Command”.
- `<get-config>` retrieves information from a either the candidate or running configuration. For more information, refer to Section 4.2, “Using the <get-config> Command”.

When getting information, you specify the information to retrieve by stating the path through the ROXII data model. You can specify the path with a series of hierarchical XML elements, or with an XPath to the desired content.

To determine the path to the desired data, you can retrieve XML files containing the ROXII data model from the device.

The ROXII database is modelled using YANG, an IETF standard for NETCONF data modelling. The data model for ROXII is defined in several *YANG* files. Typically, each RUGGEDCOM namespace is defined in a single YANG file. YANG files contain structured content, but they are not in XML.

YIN files are XML versions of YANG data model files. YIN files are well-formed XML files, making them easily parseable and able to be programmatically traversed and manipulated. You can use YIN files to find the path to every data element in the ROXII data model.

For more information on the YANG data modelling language, see Internet Engineering Task Force RFC 6020 [http://tools.ietf.org/html/rfc6020].

Section 4.1

Using the <get> Command

Use the `<get>` command to retrieve information from the *running* configuration.

The `<filter>` element contains the path to the information to retrieve. You can specify the path with hierarchical XML elements, or with an XPath.

When using hierarchical elements, do the following:

- specify the `<filter>` element’s `type` attribute as **`subtree`**
- construct the path to the desired element within the `<filter>` element
- specify the namespace for the root element in the path

You can use an XPath in the `<filter>` element, instead of the hierarchical XML element structure. For information on how to use an XPath, refer to Section 4.3, “Using XPaths with <get> and <get-config>”.

The following example shows how to return the state of the Developer Log **Enabled** setting using hierarchical XML elements. In this example, the Developer Log is enabled, so the value for the **Enabled** setting is returned as `true`.

```
<rpc message-id="3010"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <get>
  <filter type="subtree">
```

```
   <admin xmlns="http://ruggedcom.com/ns/rmf_admin">
    <logging>
     <diagnostics>
      <developer-log>
       <enabled />
      </developer-log>
     </diagnostics>
    </logging>
   </admin>
  </filter>
 </get>
</rpc>]]>]]>
```

The device returns the following:

```
<?xml version="1.0" encoding="UTF-8"?>
 <rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="3010">
  <data>
   <admin xmlns="http://ruggedcom.com/ns/rmf_admin">
    <logging>
     <diagnostics>
      <developer-log>
       <enabled>true</enabled>
      </developer-log>
     </diagnostics>
    </logging>
   </admin>
  </data>
 </rpc-reply>]]>]]>
```

Section 4.2

Using the <get-config> Command

Use the `<get-config>` command to retrieve information from a *specified* configuration, either the *running* configuration or the *candidate* configuration.

The `<filter>` element contains the path to the information to retrieve. You can specify the path with hierarchical XML elements, or with an XPath.

When using hierarchical elements, do the following:

- use the `<source>` element to specify the configuration to query. The configuration can be either `<candidate/>` or `<running/>`.
- specify the `<filter>` element's `type` attribute as **`subtree`**
- construct the path to the desired element within the `<filter>` element
- specify the namespace for the root element in the path

You can also use an XPath in the `<filter>` element, instead of the hierarchical XML element structure. For information on how to use an XPath, refer to Section 4.3, "Using XPaths with <get> and <get-config>".

The following example shows how to return the list of users from the running configuration:

```
<rpc message-id="3050"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <get-config>
  <source>
   <running/>
  </source>
  <filter type="subtree">
   <admin xmlns="http://ruggedcom.com/ns/rmf_admin">
```

```
    <users />
   </admin>
  </filter>
 </get-config>
</rpc>]]>]]>
```

The device returns the following:

```
<?xml version="1.0" encoding="UTF-8"?>
 <rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="3050">
  <data>
   <admin xmlns="http://ruggedcom.com/ns/rmf_admin">
    <users>
     <userid>
      <name>admin</name>
      <password>$1$z6HPcW$nIHgNp6EXWzN.l9SlhAVE1</password>
      <role>administrator</role>
     </userid>
     <userid>
      <name>guest</name>
      <password>$1$YEkflk$EEEV0mzCClp9oFVWVAiba1</password>
      <role>guest</role>
     </userid>
     <userid>
      <name>oper</name>
      <password>$1$1iSWr$S64yY2AzheWLDhSdbwfQP0</password>
      <role>operator</role>
     </userid>
    </users>
   </admin>
  </data>
 </rpc-reply>]]>]]>
```

Section 4.3

Using XPaths with <get> and <get-config>

Instead of using a structure of hierarchical XML elements, you can use XPaths with the `<get>` and `<get-config>` commands. XPaths are easy to construct and remove the need to specify the namespace of the data path's root element in the query.

When using an XPath, do the following:

- specify the `<filter>` element's `type` attribute as **`xpath`**.
- specify the `<filter>` element's `select` attribute with the XPath to the desired element.

The following example shows how to return the state of the Developer Log **Enabled** setting using an XPath. In this example, the Developer Log is enabled, so the value for the **Enabled** setting is returned as `true`.

```
<rpc message-id="3020"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <get>
  <filter type="xpath" select="admin/logging/diagnostics/developer-log/enabled"/>
 </get>
</rpc>]]>]]>
```

The device returns the following:

```
<?xml version="1.0" encoding="UTF-8"?>
 <rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="3020">
  <data>
   <admin xmlns="http://ruggedcom.com/ns/rmf_admin">
    <logging>
```

```
     <diagnostics>
      <developer-log>
       <enabled>true</enabled>
      </developer-log>
     </diagnostics>
    </logging>
   </admin>
  </data>
 </rpc-reply>]]>]]>
```

The following example shows how to use an XPath with the `<get-config>` command.

```
<rpc message-id="3020"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <get-config>
  <source>
   <running/>
  </source>
  <filter type="xpath" select="admin/logging/diagnostics/developer-log/enabled"/>
 </get-config>
</rpc>]]>]]>
```

The device returns the following:

```
<?xml version="1.0" encoding="UTF-8"?>
 <rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="3020">
  <data>
   <admin xmlns="http://ruggedcom.com/ns/rmf_admin">
    <logging>
     <diagnostics>
      <developer-log>
       <enabled>true</enabled>
      </developer-log>
     </diagnostics>
    </logging>
   </admin>
  </data>
 </rpc-reply>]]>]]>
```

Section 4.4

Getting Information for a Specific Object

To retrieve information for a specific object, such as a user, an interface, or other identified object, specify the identification for the item item in the XML element hierarchy or XPath.

Section 4.4.1

Specifying Objects with Hierarchical XML Elements

When using hierarchical XML elements, enclose the identifying data within its element. For example:

```
{element}
 ...
  {element}{data}{/element}
 ...
{/element}
```

Where:

- `{element}` is an element in the data model.

- `...` represents multiple elements in the data model to the target element.
- `{data}` is the identifying data for the object whose information you want to retrieve.

For example, to return the role for a specific user, send an rpc-message similar to the following:

```
<rpc message-id="3030"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <get>
  <filter type="subtree">
   <admin xmlns="http://ruggedcom.com/ns/rmf_admin">
    <users>
     <userid>
  <name>oper</name>
     <role />
     </userid>
    </users>
   </admin>
  </filter>
 </get>
</rpc>]]>]]>
```

The device returns the following rpc-reply:

```
<?xml version="1.0" encoding="UTF-8"?>
 <rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="3020">
  <data>
   <admin xmlns="http://ruggedcom.com/ns/rmf_admin">
    <users>
     <userid>
      <name>oper</name>
  <role>operator</role>
     </userid>
    </users>
   </admin>
  </data>
 </rpc-reply>]]>]]>
```

Section 4.4.2

Specifying Objects with XPaths

When using XPath, use the following syntax in the `select` statement:

```
select="{element}/.../{element}[{element}='{data}']/{element}"/>
```

- `{element}` is an element in the data model.
- `...` represents multiple elements in the data model to the target element.
- `{data}` is the identifying data for the object whose information you want to retrieve.

For example, to return the role for a specific user, send an rpc-message similar to the following:

```
<rpc message-id="3030"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" with-defaults="true">
 <get>
  <filter type="xpath"
 select="admin/users/userid[name='oper']/role"/>
 </get>
</rpc>]]>]]>
```

The device returns the following:

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

```
 <rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="3030">
  <data>
   <admin xmlns="http://ruggedcom.com/ns/rmf_admin">
    <users>
     <userid>
      <name>oper</name>
 <role>operator</role>
     </userid>
    </users>
   </admin>
  </data>
 </rpc-reply>]]>]]>
```

Section 4.5

Getting Default Values

The NETCONF standard does not require NETCONF servers to return the default values from the database. If you query an object that has a default value, the server may return just the data structure and not the default value.

For example, the NETCONF Trace Log **Enabled** setting is disabled by default. When you query this value, NETCONF does not return the default setting. When you issue this query:

```
<rpc message-id="3030"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <get>
  <filter type="xpath" select="admin/logging/diagnostics/xpath-trace-log/enabled" />
 </get>
</rpc>]]>]]>
```

The device returns the following:

```
<?xml version="1.0" encoding="UTF-8"?>
 <rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="3030">
  <data>
   <admin xmlns="http://ruggedcom.com/ns/rmf_admin">
    <logging>
     <diagnostics>
   <xpath-trace-log>
      </xpath-trace-log>
     </diagnostics>
    </logging>
   </admin>
  </data>
 </rpc-reply>]]>]]>
```

Note that the `<enabled>` element is not returned.

To return the default values for elements, add a `with-defaults` attribute to the `<rpc>` element, and set the attribute to **`true`**.

When you issue this query:

```
<rpc message-id="3030"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" with-defaults="true">
 <get>
  <filter type="xpath" select="admin/logging/diagnostics/xpath-trace-log/enabled" />
 </get>
</rpc>]]>]]>
```

The device returns the following:

```
<?xml version="1.0" encoding="UTF-8"?>
 <rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="3030">
  <data>
   <admin xmlns="http://ruggedcom.com/ns/rmf_admin">
    <logging>
     <diagnostics>
      <xpath-trace-log>
   <enabled>false</enabled>
      </xpath-trace-log>
     </diagnostics>
    </logging>
   </admin>
  </data>
 </rpc-reply>]]>]]>
```

Section 4.6

Getting Data Models from the Device

In general, all data available in the ROXII Web Interface and CLI can be accessed with NETCONF.

To get and set configuration data, you need to describe the path to the element that you want to retrieve or change. To find these paths, refer to the following references:

- the Web Interface User Guide for your device
- the RuggedCLI Reference Guide for your device
- the web-based management interface running on your device
- the command line interface (CLI) running on your device

You can also find this information by retrieving *YANG* and *YIN* files from ROXII. Retrieving these files provides you with reference material directly from the device, in file formats that can be programmatically parsed and transformed.

YANG files define the ROXII data model. YANG is an IETF standard for NETCONF data modelling. Typically, each RUGGEDCOM namespace is defined in one or more YANG files. YANG files contain structured content, but they are not in XML.

YIN files are XML versions of YANG data model files. YIN files are well-formed XML, making them easily parseable and able to be programmatically traversed and manipulated. You can use YIN files to find the path to every data element in the ROXII data model.

The paths to ROXII action commands can be determined from the ROXII YANG files, but the YANG files can be difficult to read. For a list of all ROXII actions and their paths, refer to Chapter 6, *ROXII Actions*.

For more information on the YANG data modelling language, see Internet Engineering Task Force RFC 6020 [http://tools.ietf.org/html/rfc6020].

This section describes how to retrieve YANG and YIN files from ROXII. This section also describes how to use **`pyang`**, an open source utility, to convert YANG and YIN files to human-readable text files that clearly illustrate the tree structure of the elements in the RUGGEDCOM namepspaces.

To retrieve YANG and YIN files from the device, you first need to retrieve a list of schemas. The list of schemas contains information needed to download each YANG or YIN file. For instructions on how to download the list of schemas, refer to Section 4.6.1, “Getting Schemas from the Device”.

After retrieving the list of schemas, you can download YANG and YIN files. For instructions on how to download these files, refer to Section 4.6.2, “Getting YIN and YANG Files from the Device”.

After downloading YANG and YIN files, you can use the open-source **pyang** utility to create human-readable text-based tree diagrams of the elements in the files. For instructions on using the **pyang** utility, refer to Section 4.6.3, "Using pyang".

Section 4.6.1
Getting Schemas from the Device

This section describes how to download a list of schemas from ROXII. This general list of schemas provides information needed to download specific schemas from the device.

To get a list of schemas from the device, do the following:

1. Log in to the device and start a NETCONF session. For instructions on how to initiate a NETCONF session, refer to Section 3.1, "Connecting to the NETCONF Service"
2. Issue this command:

```
<rpc message-id="4010" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <get>
  <filter type="subtree">
   <netconf-state xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-monitoring">
    <schemas/>
   </netconf-state>
  </filter>
 </get>
</rpc>
```

   The device responds with a list of schemas.
3. Review the list and locate the `<schema>` entries for items you want to retrieve. For example, shown here are the `<schema>` entries for the `rmf_admin` namespace:

```
<schema>
 <identifier>rmf_admin</identifier>
 <version>2012-03-07</version>
 <format>yin</format>
 <namespace>http://ruggedcom.com/ns/rmf_admin</namespace>
 <location>NETCONF</location>
</schema>
<schema>
 <identifier>rmf_admin</identifier>
 <version>2012-03-07</version>
 <format>yang</format>
 <namespace>http://ruggedcom.com/ns/rmf_admin</namespace>
 <location>NETCONF</location>
</schema>
```

4. Make a note of the `<identifier>`, `<format>`, and `<version>` data. You need this information to retrieve the YIN or YANG file from the device.
5. After finding the `<identifier>`, `<format>`, and `<version>` for the schema you want to download, you can retrieve the YIN and YANG files. For instructions, refer to Section 4.6.2, "Getting YIN and YANG Files from the Device".

Section 4.6.2
Getting YIN and YANG Files from the Device

To retrieve a specific YIN or YANG file, do the following:

1. Log in to the device and start a NETCONF session. For instructions on how to initiate a NETCONF session, refer to Section 3.1, “Connecting to the NETCONF Service”
2. Issue this command:

```
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <get-schema xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-monitoring">
  <identifier>{identifier}</identifier>
  <version>{version}</version>
  <format>{format}</format>
 </get-schema>
</rpc>]]>]]>
```

   - **{identifier**
     Specifies the schema name.
   - **{version}**
     Specifies the schema version number.
   - **{format}**
     Specified the schema format: can be either yin or yang.

   For example, this command retrieves the rmf_admin YIN file:

```
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <get-schema xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-monitoring">
  <identifier>rmf_admin</identifier>
  <version>2012-03-07</version>
  <format>yin</format>
 </get-schema>
</rpc>]]>]]>
```

   The device returns the text from the specified YIN or YANG file.

Section 4.6.3

Using pyang

pyang is an open-source utility used to validate and transform YANG and YIN files. **pyang** is particularly useful for transforming YANG and YIN files into text-based output that clearly illustrated the hierarchy of the elements in the ROXII data model files.

Download pyang from the pyang project site [http://code.google.com/p/pyang/]. **pyang** is a Python-based application. If you do not already have Python installed, download it from python.org [http://www.python.org/].

This section describes how to us pyang to convert a YANG or YIN file to a text-based tree diagram of a ROXII schema.

To convert a YANG or YIN file to a text-based tree diagram, do the following:

1. Before beginning, download one or more YANG or YIN files from ROXII. For instructions on downloading schemas, refer to Section 4.6.2, “Getting YIN and YANG Files from the Device”.
2. At a command line prompt; type this command:

```
pyang {inputFile} -o {outputFile} -f tree
```

   - `{inputFile}`

     The path to and filename of the YANG or YIN file that you want to convert.
   - `{outputFile}`

     The path to and filename of the text-based tree diagram that you want to create.

For example, to convert the `rmf_services.yan` file to a text-based tree diagram, type the following command. This example assumed that you are issuing the command in the same directory as where the `rmf_services.yang` file is located.

```
pyang rmf_services.yang -o rmf_services.txt -f tree
```

pyang creates the `rmf_services.txt` file.

3. Open the output file in a text editor. This example shows a portion of the `rmf_services.yang` file rendered as a text-based tree:

```
module: rmf_services
   +--rw services
      +--rw time
         +--rw ntp
            +--rw enabled?           boolean
            +--rw server [name]
            |  +--rw name            inet:host
            |  +--rw enabled?        empty
            |  +--rw peer?           empty
            |  +--rw minpoll?        ntpPollType
            |  +--rw maxpoll?        ntpPollType
            |  +--rw iburst?         empty
            |  +--rw ntp-version?    ntpVersionType
            |  +--rw prefer?         empty
            |  +--rw key?            leafref
            .
            .
            .
```

- `rw` indicates a read-write node.
- `ro` indicates a read-only node.
- `[square braces]` indicate an identifier or name for a data object.
- text following the node name indicates the data type for the node, such as boolean, string, and so on.

Section 4.6.3.1
Using the Text-Based Tree

Refer to the text-based tree to help build paths and element references in your NETCONF commands. Use the structure shown in the text-based tree diagram to build the XML used in your NETCONF `<rpc>` messages.

For example, to enable the NTP service on a device, locate the `ntp/enabled` field in the tree:

```
+--rw services
   +--rw time
      +--rw ntp
         +--rw enabled?           boolean
         .
         .
         .
```

In the XML, this tree structure looks like the following:

```
<services>
 <ntp>
  <enabled></enabled>
 </ntp>
</services>
```

To set the **Enabled** field to true, the XML in your NETCONF `<rpc>` looks like the following:

```
<rpc message-id="233" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <edit-config>
  <target>
   <candidate/>
  </target>
  <config>
    <services xmlns="http://ruggedcom.com/ns/rmf_services">
    <ntp>
     <enabled>true</enabled>
    </ntp>
   </services>
  </config>
</edit-config>
</rpc>]]>]]>
```

Note that you need to add the XML namespace to the root element in the data structure.

To address an identified object, you need to refer to the object's identifying name or key. In this example, we want to set a peer IP address for the NTP server named *ntp_server_01*.

Again, refer to the text-based tree diagram to locate the `services/ntp/server(name)/peer` field in the tree:

```
+--rw services
   +--rw time
      +--rw ntp
         +--rw enabled?           boolean
         +--rw server [name]
         |  +--rw name            inet:host
         |  +--rw enabled?        empty
         |  +--rw peer?           empty
         .
         .
         .
```

In the XML, this tree structure looks like the following:

```
<services>
 <ntp>
  <server>
   <name></name>
   <peer></peer>
  </server>
 </ntp>
</services>
```

To set a peer for the NTP server, the XML in your NETCONF `rpc` looks like the following:

```
<rpc message-id="233" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <edit-config>
  <target>
   <candidate/>
  </target>
  <config>
    <services xmlns="http://ruggedcom.com/ns/rmf_services">
    <ntp>
     <server>
      <name>ntp_server_01</name>
      <peer>192.168.0.100</peer>
     </server>
    </ntp>
   </services>
  </config>
</edit-config>
</rpc>]]>]]>
```

5 Changing Configuration Data

This section describes how to change configuration data and perform actions on your device through NETCONF.

You can edit configuration data in two ways:

- You can edit the running configuration directly. In this approach, any changes you make to the running configuration take effect immediately. You do not need to use the `<commit>` command to apply the changes. You cannot use the `<validate>` command to check the syntax of the changes before they take effect. However, you can use the `<validate>` command after making the changes to confirm that they are correct. Obviously, making changes to the running configuration is potentially risky. It is recommended that you use this approach only after gaining experience with NETCONF and confirming that your scripts and procedures work reliably.
- You can edit the candidate configuration and then commit the changes to the running configuration. In this approach, you make changes to a safe workspace called the *candidate* configuration. After making changes, you can use the `<validate>` command to confirm the syntax of the candidate configuration. If necessary, you can discard the changes with the `<discard-changes>` command, allowing you to cancel the editing process and clear any errors. After reviewing and validating the changes, you apply the changes to the running configuration with the `<commit>` command. Editing the candidate configuration and then committing the changes is the recommended approach for editing configuration data.

Section 5.1

Changing Data in the Running Configuration

To edit data in the running configuration, simply specify the running configuration as the target in the edit-config command. You do not need to commit the change with the commit command. You cannot validate the syntax of the change with the validate command before the change takes effect. If you want to validate the change, use the validate commands on the running configuration after making the change.

CAUTION!
Making changes directly to the running configuration is potentially risky: you may interrupt service on the device, or your changes may conflict with those of other users working on other management interfaces. Making changes to the running configuration should only be done by those with sufficient system expertise and experience to make sure that such changes are performed properly.

To edit the running configuration, do the following:

1. Connect to and log in to the device.
2. Issue the `<edit-config>` command:

```
<rpc message-id="233" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <edit-config>
  <target>
   <running/>
  </target>
  <config>
  {path}
  </config>
 </edit-config>
</rpc>]]>]]>
```

- `{path}`

  The XML elements describing the path to and the data for the element to be edited.

For example, to create a static VLAN:

```
<rpc message-id="233" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <edit-config>
  <target>
   <running/>
  </target>
  <config>
   <switch xmlns="http://ruggedcom.com/ns/rmf_ifswitch">
    <vlans>
     <static-vlan>
      <vid>0086</vid>
     </static-vlan>
    </vlans>
   </switch>
  </config>
 </edit-config>
</rpc>]]>]]>
```

After making changes to the running configuration, you can use the validate command to check the syntax on the configuration. For instructions on how to validate a configuration, refer to Section 5.2.5, “Validating Changes”.

Section 5.2

Changing Data in the Candidate Configuration

The recommended approach for changing data is to make your changes to the candidate configuration before committing the changes to the running configuration. Making changes to the candidate configuration provides the opportunity to validate the syntax of the configuration and to discard the changes if required.

As illustrated in Section 1.3.3, “Sample Session: Editing Data”, editing data in the candidate configuration and committing it involves a multi-step workflow:

1. Connect to the device and say hello.
2. Lock the candidate and running configuration datastores.
3. Discard any stray changes to restore the configurations to a known state.
4. Edit the candidate configuration.
5. Validate the candidate configuration.
6. Commit the changes.
7. Unlock the datastores.
8. Close the session.

This section describes how to lock the datastores, edit and delete data, validate the configuration, commit the changes, and lock the datastores.

For instructions on how to initiate a NETCONF session, refer to Section 3.1, “Connecting to the NETCONF Service”. For instructions on how to close a NETCONF session, refer to Section 3.3, “Closing the Session”.

Section 5.2.1
Locking Data Stores

To lock the candidate and running datastores, do the following:

1. Issue an <rpc> request to lock the running configuration:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1010">
 <lock>
  <target>
   <running/>
  </target>
 </lock>
</rpc>
]]>]]>
```

   - All commands must be enclosed within <rpc> tags. The message-id attribute is not required but is recommended. The message-id attribute is returned in the device response, allowing you to match responses with requests.
   - The <lock> element indicates that this request is to lock a configuration.
   - The <target> element specifies the configuration to lock. In this <rpc>, the lock target is the <running/> configuration.
   - The]]>]]> string indicates the end of the NETCONF message. Each NETCONF message must end with]]>]]>

   The device responds with the following:

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1010">
 <ok/>
</rpc-reply>]]>]]>
```

   The running configuration is now locked

2. Issue an <rpc> request to lock the candidate configuration:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1011">
 <lock>
  <target>
   <candidate/>
  </target>
 </lock>
</rpc>]]>]]>
```

   The device responds with the following:

```
<?xml version="1.0" encoding="UTF-8"?>
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="1011">
 <ok/>
</rpc-reply>]]>]]>
```

   The candidate configuration is now locked.

Section 5.2.2
Copying Data

You can use the copy-config command to do the following:

- copy a specified configuration to an XML file on the device. Do this when you want to save a configuration to a file and then download the file through the web interface or command line interface.
- copy an XML file on the device to a specified configuration. Do these when you want to overwrite a specified configuration with a file that has been uploaded to the device through the web interface or through the command line interface.

When using copy-config to save the configuration to an XML file, the file is saved in the /var/lib/config directory on the device. You can download the through the ROXII web interface or through the command line interface.

To save a configuration to an XML file, do the following:

1. Connect to and log in to the device.
2. Issue the copy-config command:

```
<rpc message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <copy-config>
  <target>
   <url>file:///{filename.xml}</url>
  </target>
  <source>
   <{configuration}/>
  </source>
 </copy-config>
</rpc>]]>]]>
```

- `{filename.xml}`

  Specify a filename for the new XML file.

- `{configuration}`

  Specify the configuration to save: running or candidate.

  For example, this command saves the running configuration to a file named `running_config_01.xml`:

```
<rpc message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <copy-config>
  <target>
   <url>file:///running_config_01.xml</url>
  </target>
  <source>
   <running/>
  </source>
 </copy-config>
</rpc>]]>]]>
```

  The file is saved to the /var/lib/config directory on the device.

To overwrite a configuration with a configuration file, do the following:

1. Upload an XML configuration file to the device. For instructions on how to upload a configuration file, refer to Appendix A, *Uploading Files to the Device*.
2. Connect to and log in to the device.
3. Lock the running configuration:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="230">
 <lock>
  <target>
   <running/>
  </target>
 </lock>
</rpc>]]>]]>
```

4. Lock the target configuration:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="231">
 <lock>
  <target>
   <candidate/>
  </target>
 </lock>
</rpc>]]>]]>
```

5. Discard any configuration changes:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="232">
 <discard-changes/>
</rpc>]]>]]>
```

6. Use copy-config to copy the file to a specified configuration:

```
<rpc message-id="233"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <copy-config>
  <target>
   <{configuration}/>
  </target>
  <source>
   <url>file:///{filename}</url>
  </source>
 </copy-config>
</rpc>]]>]]>
```

- `{configuration}`

  The target configuration: can be candidate or running.

- `{filename}`

  The name of the configuration file uploaded to /var/lib/config

For example, this command loads the configuration file `standard_config.xml` to the running configuration:

```
<rpc message-id="233"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <copy-config>
  <target>
   <running/>
  </target>
  <source>
   <url>file:///standard_config.xml</url>
  </source>
 </copy-config>
</rpc>]]>]]>
```

7. Commit the changes:

```
<rpc message-id="234" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <commit/>
</rpc>]]>]]>
```

8. Unlock the candidate configuration:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="235">
 <unlock>
  <target>
   <candidate/>
  </target>
 </unlock>
```

```
</rpc>]]>]]>
```

9. Unlock the running configuration:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="236">
 <unlock>
  <target>
   <running/>
  </target>
 </unlock>
</rpc>]]>]]>
```

Section 5.2.3

Replacing Data

To replace an existing configuration value with a new value, do the following:

1. Connect to and log in to the device.
2. Lock the running configuration:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="230">
 <lock>
  <target>
   <running/>
  </target>
 </lock>
</rpc>]]>]]>
```

3. Lock the target configuration:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="231">
 <lock>
  <target>
   <candidate/>
  </target>
 </lock>
</rpc>]]>]]>
```

4. Discard any configuration changes:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="232">
 <discard-changes/>
</rpc>]]>]]>
```

5. Issue an `<rpc>` request with the replace operation:

```
<rpc message-id="233" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <edit-config>
  <target>
   <candidate/>
  </target>
  <config>
   <{root element}
       xmlns="{namespace URL}"
       xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0">
    {configuration data with nc:operation="replace" attribute}
   </{root element}>
  </config>
 </edit-config>
</rpc>]]>]]>
```

- `{root element}`

  The top level element in the data model under which the data is located. Note that you need to declare the `xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0"` namespace at this point.

- `{namespace}`

  The URL to the RUGGEDCOM namespace for the top level element.

- `{configuration data with nc:operation="replace" attribute}`

  The path to the data to be replaced, with the `nc:operation="replace"` attribute on the element containing the data to be replaced.

For example, to replace an existing IP address with a new address, issue the following request.

```
<rpc message-id="233" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <edit-config>
  <target>
   <candidate/>
  </target>
  <config>
   <ip xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0"
       xmlns="http://ruggedcom.com/ns/rmf_ip">
    <ifname>fe-cm-1</ifname>
    <ipv4 nc:operation="replace">
     <address>
      <ipaddress>192.168.1.42/24</ipaddress>
     </address>
    </ipv4>
   </ip>
  </config>
 </edit-config>
</rpc>]]>]]>
```

6. Commit the change:

```
<rpc message-id="234" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <commit/>
</rpc>]]>]]>
```

7. Unlock the candidate configuration:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="235">
 <unlock>
  <target>
   <candidate/>
  </target>
 </unlock>
</rpc>]]>]]>
```

8. Unlock the target configuration:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="236">
 <unlock>
  <target>
   <running/>
  </target>
 </unlock>
</rpc>]]>]]>
```

Section 5.2.4

Deleting Data

To delete a specific configuration setting, do the following:

1. Connect to and log in to the device.
2. Lock the running configuration:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="230">
 <lock>
  <target>
   <running/>
  </target>
 </lock>
</rpc>]]>]]>
```

3. Lock the target configuration:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="231">
 <lock>
  <target>
   <candidate/>
  </target>
 </lock>
</rpc>]]>]]>
```

4. Discard any configuration changes:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="232">
 <discard-changes/>
</rpc>]]>]]>
```

5. Issue an `<rpc>` request with the delete operation:

```
<rpc message-id="233" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <edit-config>
  <target>
   <candidate/>
  </target>
  <config>
   <{root element}
       xmlns="{namespace URL}"
       xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0">
    {configuration data with nc:operation="delete" attribute}
   </{root element}>
  </config>
 </edit-config>
</rpc>]]>]]>
```

- `{root element}`

  The top level element in the data model under which the data is located. Note that you need to declare the `xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0"` namespace at this point.

- `{namespace}`

  The URL to the RUGGEDCOM namespace for the top level element.

- `{configuration data with nc:operation="delete" attribute}`

  The path to the data to be deleted, with the `nc:operation="delete"` attribute on the element containing the data to be deleted.

For example, to clear the **Passive Interface** setting on an interface in OSPF, issue the following request.

```
<rpc message-id="233" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <edit-config>
  <target>
   <candidate/>
  </target>
  <config>
   <routing xmlns="http://ruggedcom.com/ns/rmf_routing"
               xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0">
    <dynamic>
     <ospf>
      <interface>
       <ifname>switch.0022</ifname>
        <passive nc:operation="delete"/>
      </interface>
     </ospf>
    </dynamic>
   </routing>
  </config>
 </edit-config>
</rpc>]]>]]>
```

6. Commit the change:

```
<rpc message-id="234" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <commit/>
</rpc>]]>]]>
```

7. Unlock the candidate configuration:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="235">
 <unlock>
  <target>
   <candidate/>
  </target>
 </unlock>
</rpc>]]>]]>
```

8. Unlock the target configuration:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="236">
 <unlock>
  <target>
   <running/>
  </target>
 </unlock>
</rpc>]]>]]>
```

Section 5.2.5

Validating Changes

You can validate the syntax of a specified configuration with the validate comment. Validation confirms the syntax of the specified configuration. After making extensive changes to the candidate configuration, it is recommended that your validate the candidate configuration before committing it.

To validate a configuration, do the following:

1. Connect to and log in to the device.
2. Issue an `<rpc>` request with the validation command:

```
<rpc message-id="103"
 xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
```

```
  <validate>
    <source>
      <{configuration}/>
    </source>
  </validate>
</rpc>]]>]]>
```

- `{configuration}`

  The configuration to validate: candidate or running.

For example, to validate the candidate configuration, issue this command:

```
<rpc message-id="103"
 xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <validate>
    <source>
      <candidate/>
    </source>
  </validate>
</rpc>]]>]]>
```

If the configuration syntax is correct, the device responds with the following:

```
<?xml version="1.0" encoding="UTF-8"?>
 <rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="103">
  <ok/>
 </rpc-reply>]]>]]>
```

If the configuration syntax is not correct, the device responds with an `<rpc-error>` message. For example:

```
<?xml version="1.0" encoding="UTF-8"?>
 <rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="103">
  <rpc-error>
   <error-type>application</error-type>
   <error-tag>operation-failed</error-tag>
   <error-severity>error</error-severity>
   <error-path xmlns:rmf_admin="http://ruggedcom.com/ns/rmf_admin">
    /rmf_admin:admin/rmf_admin:authentication
   </error-path>
   <error-message xml:lang="en">/admin/authentication: admin/timezone must be set</error-message>
   <error-info>
    <bad-element>authentication</bad-element>
   </error-info>
  </rpc-error>
 </rpc-reply>]]>]]>
```

- The `<error-type>`, `<error-tag>`, and `<error-severity>` elements provide information about the nature of the syntax error.
- The `<error-path>` element indicates where in the configuration the syntax error is found.
- The `<error-message>` element provides a message, when one is available, describing the error.
- The `<bad-element>` element indicates the element related to the error.

For more information on NETCONF errors, see Internet Engineering Task Force RFC 6241 Appendix A. NETCONF Error List [http://tools.ietf.org/html/rfc6241#appendix-A].

Section 5.2.6

Committing Changes

After making changes to the candidate configuration, you can commit the changes to make the changes active in the running configuration. It is recommended that you first validate the candidate configuration before issuing the commit command. For instructions on how to validate a configuration, refer to Section 5.2.5, “Validating Changes”.

To commit changes made to the candidate configuration, issue this command:

```
<rpc message-id="234" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <commit/>
</rpc>]]>]]>
```

6 ROXII Actions

This section describes how to activate ROXII actions on the device with an <rpc> message through NETCONF. Actions perform functions directly on the device, such as shutting down the device, rebooting, clearing port statistics, and more.

To activate most actions, the <rpc> simply specifies the path to the action in the ROXII data model. Some actions require parameters, such as module and port numbers, object identifiers, and other other settings. Where required, this section describes the parameters for each action.

This section organizes the actions by the RUGGEDCOM namespace in which the actions are found.

Table: Actions and ROXII Web Interface Equivalents

Action	ROXII Web UI Equivalent
Section 6.1.1, “snmp-discover”	/admin/snmp/snmp-discover
Section 6.1.2, “launch-upgrade”	/admin/software-upgrade/launch-upgrade
Section 6.1.3, “decline-upgrade”	/admin/software-upgrade/decline-upgrade
Section 6.1.4, “rollback-reboot”	/admin/software-upgrade/rollback-reboot
Section 6.1.5, “roxflash”	/admin/rox-imaging/roxflash
Section 6.1.6, “clear-all-alarms”	/admin/clear-all-alarms
Section 6.1.7, “acknowledge-all-alarms”	/admin/acknowledge-all-alarms
Section 6.1.8, “shutdown”	/admin/shutdown
Section 6.1.9, “reboot”	/admin/reboot
Section 6.1.10, “set-system-clock”	/admin/set-system-clock
Section 6.1.11, “restore-factory-defaults”	/admin/restore-factory-defaults
Section 6.1.12, “delete-logs”	/admin/delete-logs
Section 6.1.13, “install-files”	/admin/install-files
Section 6.1.14, “backup-files (Backup Files)”	/admin/install-files
Section 6.1.15, “full-configuration-save”	/admin/full-configuration-save
Section 6.1.16, “full-configuration-load”	/admin/full-configuration-load
Section 6.2.1, “clearstatistics (WAN interfaces)”	/interfaces/wan/clearstatistics
Section 6.2.2, “loopback (WAN interfaces)”	/interfaces/wan/loopback
Section 6.2.3, “reset (Modem)”	/interfaces/modem{identifier}/reset
Section 6.2.4, “at (Modem)”	/interfaces/modem{identifier}/at
Section 6.2.5, “reset (Cellular Modem)”	/interfaces/cellmodem{identifier}/reset
Section 6.2.6, “at (Cellular Modem)”	/interfaces/cellmodem{identifier}/at
Section 6.2.7, “reset (Serial Port)”	/interfaces/serial/port{interface}/reset
Section 6.2.8, “clear-serial-port-stats”	/interfaces/serial/port{interface}/clear-serial-port-stats

Action	ROXII Web UI Equivalent
Section 6.2.9, “restart-serserver”	/interfaces/serial/restart-serserver
Section 6.2.10, “reset-port (Switch Port)”	/interfaces/switch{interface}/reset-port
Section 6.2.11, “clear-port-stats (Switch Port)”	/interfaces/switch{interface}/clear-port-stats
Section 6.2.12, “start-cable-test (Switch Port)”	interfaces/switch{interface}/diagnostics/start-cable-test
Section 6.2.13, “clear-cable-stats-port (Switch Port)”	/interfaces/switch{interface}/diagnostics/clear-cable-stats-port
Section 6.3.1, “ntp-status”	/services/time/ntp/ntp-status
Section 6.3.2, “log (Link-Failover)”	/services/link-failover{interface}/log
Section 6.3.3, “start-test (Link Failover)”	/services/link-failover{interface}/start-test
Section 6.3.4, “cancel-test (Link Failover)”	/services/link-failover{interface}/cancel-test
Section 6.3.5, “show-active-leases (DHCP Server)”	/services/dhcpserver/show-active-leases
Section 6.4.1, “clear-stp-stats (Switch)”	/switch/spanning-tree/clear-stp-stats
Section 6.4.2, “flush-dynamic-rules (Switch)”	/switch/layer3-switching/flush-dynamic-rules
Section 6.4.3, “reset-all-switch-ports (Switch)”	/switch/reset-all-switch-ports
Section 6.4.4, “clear-all-switch-stats (Switch)”	/switch/clear-all-switch-stats
Section 6.4.5, “clear-cable-stats-all (Switch)”	/switch/clear-cable-stats-all

Section 6.1
Admin Namespace Actions

Section 6.1.1
snmp-discover

This action discovers the SNMP engine ID for a given IP address and port. Parameters include the `{ip address}`, `{port}`, and `{trap-port}`.

```
<rpc message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <action xmlns="http://tail-f.com/ns/netconf/actions/1.0">
  <data>
   <admin xmlns="http://ruggedcom.com/ns/rmf_admin">
    <snmp>
     <snmp-discover>
      <address>{ipAaddress}</address>
      <port>{port}</port>
      <trap-port>{trapPort}</trap-port>
     </snmp-discover>
    </snmp>
   </admin>
  </data>
 </action>
</rpc>]]>]]>
```

- `{ipAddress}`

The SNMP IP address the device listens on.

- `{port}`

  The SNMP data port the device listens on (if any).

- `{trapPort}`

  The SNMP trap port the device listens on (if any).

Section 6.1.2
launch-upgrade

This action launches a ROXII software upgrade to the alternate partition on the device. The repository address and target release must be configured in `admin/software-upgrade/upgrade-settings`. A reboot is required to run the new software release in the alternate partition. All configurations are locked from the start of the upgrade to the subsequent reboot.

This action does not take any parameters.

```
<rpc message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <action xmlns="http://tail-f.com/ns/netconf/actions/1.0">
  <data>
   <admin xmlns="http://ruggedcom.com/ns/rmf_admin">
    <software-upgrade>
     <launch-upgrade/>
    </software-upgrade>
   </admin>
  </data>
 </action>
</rpc>]]>]]>
```

Section 6.1.3
decline-upgrade

This action declines a ROXII software upgrade. After an upgrade occurs and while the system is awaiting a reboot to the upgraded partition, use the `<decline-upgrade>` action to cancel the attempt to run the upgraded partition. This action also unlocks all configurations locked by the `<launch-upgrade>` process. If no update has been applied, or if the device is not awaiting a reboot after applying an update, this action has no effect.

This action does not take any parameters.

```
<rpc message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <action xmlns="http://tail-f.com/ns/netconf/actions/1.0">
  <data>
   <admin xmlns="http://ruggedcom.com/ns/rmf_admin">
    <software-upgrade>
     <decline-upgrade/>
    </software-upgrade>
   </admin>
  </data>
 </action>
</rpc>]]>]]>
```

Section 6.1.4
rollback-reboot

This action boots the device to a previous software release on the alternate partition. This action does not take any parameters.

```
<rpc message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <action xmlns="http://tail-f.com/ns/netconf/actions/1.0">
  <data>
   <admin xmlns="http://ruggedcom.com/ns/rmf_admin">
    <software-upgrade>
     <rollback-reboot/>
    </software-upgrade>
   </admin>
  </data>
 </action>
</rpc>]]>]]>
```

Section 6.1.5
roxflash

This action flashes a ROXII image to the alternate partition. On rebooting, the device boots from the flashed partition. Configurations are not transferred. This action takes a single parameter: `{url}`.

```
<rpc message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <action xmlns="http://tail-f.com/ns/netconf/actions/1.0">
  <data>
   <admin xmlns="http://ruggedcom.com/ns/rmf_admin">
    <rox-imaging>
     <roxflash>
      <url>{url}</url>
     </roxflash>
    </rox-imaging>
   </admin>
  </data>
 </action>
</rpc>]]>]]>
```

- `{url}`

  The URL of the ROXII image to download. The file transfer supports SFTP, FTP, and HTTP. The URL format is `protocol://user:password@host:port/path-to-file`. If your server does not require authentication, you may omit `user:password`. When using the default port for the protocol, you may omit `:port`.

Section 6.1.6
clear-all-alarms

This action clears all clearable alarms in the active list. Note that not all alarms can be cleared. This action does not take any parameters.

```
<rpc message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <action xmlns="http://tail-f.com/ns/netconf/actions/1.0">
  <data>
```

```
   <admin xmlns="http://ruggedcom.com/ns/rmf_admin">
    <clear-all-alarms/>
   </admin>
  </data>
 </action>
</rpc>]]>]]>
```

Section 6.1.7
acknowledge-all-alarms

This action acknowledges all alarms in the active list. The alarms remain in the active list, but Alarm LED and critical alarm relay are shut off. This action does not take any parameters.

```
<rpc message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <action xmlns="http://tail-f.com/ns/netconf/actions/1.0">
  <data>
   <admin xmlns="http://ruggedcom.com/ns/rmf_admin">
    <acknowledge-all-alarms/>
   </admin>
  </data>
 </action>
</rpc>]]>]]>
```

Section 6.1.8
shutdown

This action shuts down the device. After using this action, the device shuts down and provides a time-out period during which you can remove power from the device. The default time-out period is 300 seconds (five minutes). At the end of the time-out period, the device reboots and restarts.

This action does not take any parameters.

```
<rpc message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <action xmlns="http://tail-f.com/ns/netconf/actions/1.0">
  <data>
   <admin xmlns="http://ruggedcom.com/ns/rmf_admin">
    <shutdown/>
   </admin>
  </data>
 </action>
</rpc>]]>]]>
```

Section 6.1.9
reboot

This action reboots the device. This action does not take any parameters.

```
<rpc message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <action xmlns="http://tail-f.com/ns/netconf/actions/1.0">
  <data>
   <admin xmlns="http://ruggedcom.com/ns/rmf_admin">
```

```
    <reboot/>
   </admin>
  </data>
 </action>
</rpc>]]>]]>
```

Section 6.1.10
set-system-clock

This action sets the date and time on the system. This action takes a single parameter: `<time>`.

```
<rpc message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <action xmlns="http://tail-f.com/ns/netconf/actions/1.0">
  <data>
   <admin xmlns="http://ruggedcom.com/ns/rmf_admin">
    <set-system-clock>
     <time>{time}</time>
    </set-system-clock>
   </admin>
  </data>
 </action>
</rpc>]]>]]>
```

- `{time}`

  The date and time in the format YYYY-MM-DD HH:MM:SS.

Section 6.1.11
restore-factory-defaults

This action restores the device to its factory default settings. This action does not take any parameters.

```
<rpc message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <action xmlns="http://tail-f.com/ns/netconf/actions/1.0">
  <data>
   <admin xmlns="http://ruggedcom.com/ns/rmf_admin">
    <restore-factory-defaults/>
   </admin>
  </data>
 </action>
</rpc>]]>]]>
```

Section 6.1.12
delete-logs

This action deletes all log files on the device. This action does not take any parameters.

```
<rpc message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <action xmlns="http://tail-f.com/ns/netconf/actions/1.0">
  <data>
   <admin xmlns="http://ruggedcom.com/ns/rmf_admin">
    <delete-logs/>
```

```
   </admin>
  </data>
 </action>
</rpc>]]>]]>
```

Section 6.1.13
install-files

This action copies files from a specified URL to the device. Parameters include `<file-type>` and `<url>`.

```
<rpc message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <action xmlns="http://tail-f.com/ns/netconf/actions/1.0">
  <data>
   <admin xmlns="http://ruggedcom.com/ns/rmf_admin">
    <install-files>
     <file-type>{fileType}</file-type>
     <url>{url}</url>
    </install-files>
   </admin>
  </data>
 </action>
</rpc>]]>]]>
```

- `{fileType}`

  The type of file to copy to the device. Must be one of the following: **`config`**, **`featurekey`**, **`elancertificate`**, **`ipseccertificate`**, **`cacertificate`**, or **`crlfiles`**.

- `{url}`

  The URL and filename of the ROXII file to copy. The file transfer supports SCP, SFTP, FTPS, and HTTP. The URL format is `protocol://user:password@host:port/path-to-file`. If the port is not specified, the device uses the default port for the protocol.

Section 6.1.14
backup-files (Backup Files)

This action copies files from the device to a specified URL. Parameters include `<file-type>`, `<file>`, `<timestamp>`, and `<url>`.

```
<rpc message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <action xmlns="http://tail-f.com/ns/netconf/actions/1.0">
  <data>
   <admin xmlns="http://ruggedcom.com/ns/rmf_admin">
    <backup-files>
     <file-type>{fileType}</file-type>
     <file>{file}</file>
     <timestamp>{timeStamp}</timestamp>
     <url>{url}</url>
    </backup-files>
   </admin>
  </data>
 </action>
</rpc>]]>]]>
```

- `{fileType}`

The type of file to copy from the device. Must be one of the following: **config**, **featurekey**, **elancertificate**, **ipseccertificate**, **cacertificate**, or **crlfiles**.

- {file}

  The name of the file to copy.

- {timeStamp}

  A Boolean value: **true** or **false**. When **true**, the system appends a timestamp to the filename. This option does not apply if the file name contains an * (asterisk) character.

- {url}

  The URL to which to copy the file. The file transfer supports SCP, SFTP, FTP, and HTTP. The URL format is protocol://user:password@host:port/path-to-file/. Note that the URL must end with a / (forward slash_ character. If the port is not specified, the device uses the default port for the protocol.

Section 6.1.15
full-configuration-save

This action saves the ROXII configuration in the specified format to a specified file. Files are saved to the /var/lib/config directory on the device. Parameters include <format> and <file-name>.

```
<rpc message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <action xmlns="http://tail-f.com/ns/netconf/actions/1.0">
  <data>
   <admin xmlns="http://ruggedcom.com/ns/rmf_admin">
    <full-configuration-save>
     <format>{format}</format>
     <file-name>{fileName}</file-name>
    </full-configuration-save>
   </admin>
  </data>
 </action>
</rpc>]]>]]>
```

- {format}

  The format for the configuration file: **cli** or **netconf**.

- {fileName}

  The name for the configuration file.

Section 6.1.16
full-configuration-load

This action loads a configuration from the specified file found in the /var/lib/config directory on the device. Parameters include <format> and <file-name>.

```
<rpc message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <action xmlns="http://tail-f.com/ns/netconf/actions/1.0">
  <data>
   <admin xmlns="http://ruggedcom.com/ns/rmf_admin">
    <full-configuration-load>
     <format>{format}</format>
```

```
     <file-name>{fileName}</file-name>
    </full-configuration-load>
   </admin>
  </data>
 </action>
</rpc>]]>]]>
```

- `{format}`

  The format for the configuration file: **`cli`** or **`netconf`**.

- `{fileName}`

  The name for the configuration file.

Section 6.2
Interfaces Namespace Actions

Section 6.2.1
clearstatistics (WAN interfaces)

This action clears the statistics for the specified WAN interface. Parameters include `<ddsname>`, `<t1e1name>`, `<t3e3name>`, and `<all-interfaces>`.

```
<rpc message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <action xmlns="http://tail-f.com/ns/netconf/actions/1.0">
  <data>
   <interfaces xmlns="http://ruggedcom.com/ns/rmf_ifs">
    <wan>
     <clearstatistics>
      <ddsName>{ddsDame}</ddsName>
      <t1e1Name>{t1e1Name}</t1e1Name>
      <t3e3Name>{t3e3Name}</t3e3Name>
      <all-interfaces/>
     </clearstatistics>
    </wan>
   </interfaces>
  </data>
 </action>
</rpc>]]>]]>
```

- `{ddsName}`

  The name of a DDS interface for which to clear statistics.

- `{t1e1Name}`

  The name of a T1E1 interface for which to clear statistics.

- `{t3e3Name}`

  The name of a T3E3 interface for which to clear statistics.

- `{allInterfaces}`

  A Boolean value: **`true`** or **`false`**. When **`true`**, the action clears all WAN interfaces.

Section 6.2.2
loopback (WAN interfaces)

This action performs a loopback test on a specified WAN interface. Parameters include `<physical-name>`, `<type>`, `<nloops>`, and `<duration>`.

```
<rpc message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <action xmlns="http://tail-f.com/ns/netconf/actions/1.0">
  <data>
   <interfaces xmlns="http://ruggedcom.com/ns/rmf_ifs">
    <wan>
     <loopback>
      <physical-name>{physicalName}</physical-name>
      <type>{type}</type>
      <nloops>{nLoops}</nloops>
      <duration>{duration}</duration>
     </loopback>
    </wan>
   </interfaces>
  </data>
 </action>
</rpc>]]>]]>
```

- `{physicalName}`

  The physical interface name.

- `{type}`

  The loopback type.

- `{nLoops}`

  The number of loops.

- `{duration}`

  The duration of the test in seconds.

Section 6.2.3
reset (Modem)

This action resets the modem. Resetting the modem takes approximately 15 seconds. Resetting the modem terminates the PPP connection.

Specify the modem interface name in the `<ifname>` element. This action does not take any parameters.

```
<rpc message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <action xmlns="http://tail-f.com/ns/netconf/actions/1.0">
  <data>
   <interfaces xmlns="http://ruggedcom.com/ns/rmf_ifs">
    <modem>
     <ifname>{interfaceName}</ifname>
     <reset/>
    </modem>
   </interfaces>
  </data>
 </action>
</rpc>]]>]]>
```

- `{interfaceName}`

  The interface name for the modem.

Section 6.2.4
at (Modem)

This action sends an AT command to the modem. The command must begin with the prefix AT.

Specify the modem interface name in the `<ifname>` element. This action takes a single parameter: `<command>`.

```
<rpc message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <action xmlns="http://tail-f.com/ns/netconf/actions/1.0">
  <data>
   <interfaces xmlns="http://ruggedcom.com/ns/rmf_ifs">
    <modem>
     <ifname>{interfaceName}</ifname>
     <at>
      <command>{atCommand}</command>
     </at>
    </modem>
   </interfaces>
  </data>
 </action>
</rpc>]]>]]>
```

- `{interfaceName}`

  The name of the modem interface.

- `{atCommand}`

  The AT command to send to the modem. The command must begin with the prefix AT.

Section 6.2.5
reset (Cellular Modem)

This action resets the cellular modem. Specify the modem module and port in the `<module>` and `<port>` elements. This action does not take any parameters.

```
<rpc message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <action xmlns="http://tail-f.com/ns/netconf/actions/1.0">
  <data>
   <interfaces xmlns="http://ruggedcom.com/ns/rmf_ifs">
    <cellmodem>
     <module>{module}</module>
     <port>{port}</port>
     <reset/>
    </cellmodem>
   </interfaces>
  </data>
 </action>
</rpc>]]>]]>
```

- `{module}`

  The module number for the cellular modem.

- `{port}`

  The port number for the cellular modem.

Section 6.2.6
at (Cellular Modem)

This action sends an AT command to the cellular modem. The command must begin with the prefix AT. Specify the modem module and port in the `<module>` and `<port>` elements. This action takes a single parameter: `<command>`.

```
<rpc message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <action xmlns="http://tail-f.com/ns/netconf/actions/1.0">
  <data>
   <interfaces xmlns="http://ruggedcom.com/ns/rmf_ifs">
    <cellmodem>
     <module>{module}</module>
     <port>{port}</port>
     <at>
      <command>{atCommand}</command>
     </at>
    </cellmodem>
   </interfaces>
  </data>
 </action>
</rpc>]]>]]>
```

- `{module}`

  The module number for the cellular modem.

- `{port}`

  The port number for the cellular modem.

- `{atCommand}`

  The AT command to send to the modem. The command must begin with the prefix AT.

Section 6.2.7
reset (Serial Port)

This action resets the specified serial port. Specify the serial module and port in the `<module>` and `<port>` elements. This action does not take any parameters.

```
<rpc message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <action xmlns="http://tail-f.com/ns/netconf/actions/1.0">
  <data>
   <interfaces xmlns="http://ruggedcom.com/ns/rmf_ifs">
    <serial>
     <module>{module}</module>
     <port>{port}</port>
     <reset/>
    </serial>
   </interfaces>
  </data>
 </action>
```

```
</rpc>]]>]]>
```

- {module}

  The module number for the serial port.

- {port}

  The port number for the serial port.

Section 6.2.8

clear-serial-port-stats

This action clears the port statistics for the specified serial port. Specify the serial module and port in the <module> and <port> elements. This action does not take any parameters.

```
<rpc message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <action xmlns="http://tail-f.com/ns/netconf/actions/1.0">
  <data>
   <interfaces xmlns="http://ruggedcom.com/ns/rmf_ifs">
    <serial>
     <module>{module}</module>
     <port>{port}</port>
     <clear-serial-port-stats/>
    </serial>
   </interfaces>
  </data>
 </action>
</rpc>]]>]]>
```

- {module}

  The module number for the serial port.

- {port}

  The port number for the serial port.

Section 6.2.9

restart-serserver

This action restarts the serial server. This action does not take any parameters.

```
<rpc message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <action xmlns="http://tail-f.com/ns/netconf/actions/1.0">
  <data>
   <interfaces xmlns="http://ruggedcom.com/ns/rmf_ifs">
    <serial>
     <restart-serserver/>
    </serial>
   </interfaces>
  </data>
 </action>
</rpc>]]>]]>
```

Section 6.2.10
reset-port (Switch Port)

This action resets the specified switch port. Specify the switch module and port in the `<module>` and `<port>` elements. This action does not take any parameters.

```
<rpc message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <action xmlns="http://tail-f.com/ns/netconf/actions/1.0">
  <data>
   <interfaces xmlns="http://ruggedcom.com/ns/rmf_ifs">
    <switch>
     <module>{module}</module>
     <port>{port}</port>
     <reset-port/>
    </switch>
   </interfaces>
  </data>
 </action>
</rpc>]]>]]>
```

- `{module}`

  The module number for the switch port.

- `{port}`

  The port number for the switch port.

Section 6.2.11
clear-port-stats (Switch Port)

This action clears the port statistics for the specified switch port. Specify the switch module and port in the `<module>` and `<port>` elements. This action does not take any parameters.

```
<rpc message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <action xmlns="http://tail-f.com/ns/netconf/actions/1.0">
  <data>
   <interfaces xmlns="http://ruggedcom.com/ns/rmf_ifs">
    <switch>
     <slot>{module}</slot>
     <port>{port}</port>
     <clear-port-stats/>
    </switch>
   </interfaces>
  </data>
 </action>
</rpc>]]>]]>
```

- `{module}`

  The module number for the switch port.

- `{port}`

  The port number for the switch port.

Section 6.2.12
start-cable-test (Switch Port)

This action starts cable test diagnostics on the specified switch port. Specify the switch module and port in the `<module>` and `<port>` elements. This action does not take any parameters.

```
<rpc message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <action xmlns="http://tail-f.com/ns/netconf/actions/1.0">
  <data>
   <interfaces xmlns="http://ruggedcom.com/ns/rmf_ifs">
    <switch>
     <slot>{module}</slot>
     <port>{port}</port>
     <diagnostics>
      <start-cable-test/>
     </diagnostics>
    </switch>
   </interfaces>
  </data>
 </action>
</rpc>]]>]]>
```

- `{module}`

  The module number for the switch port.

- `{port}`

  The port number for the switch port.

Section 6.2.13
clear-cable-stats-port (Switch Port)

This action clears the cable test diagnostic statistics on the specified switch port. Specify the switch module and port in the `<module>` and `<port>` elements. This action does not take any parameters.

```
<rpc message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <action xmlns="http://tail-f.com/ns/netconf/actions/1.0">
  <data>
   <interfaces xmlns="http://ruggedcom.com/ns/rmf_ifs">
    <switch>
     <slot>{module}</slot>
     <port>{port}</port>
     <diagnostics>
      <clear-cable-stats-port/>
     </diagnostics>
    </switch>
   </interfaces>
  </data>
 </action>
</rpc>]]>]]>
```

- `{module}`

  The module number for the switch port.

- `{port}`

  The port number for the switch port.

Section 6.3
Services Namespace Actions

Section 6.3.1
ntp-status

This action displays the status of the running NTP system. This action does not take any parameters.

```
<rpc message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <action xmlns="http://tail-f.com/ns/netconf/actions/1.0">
  <data>
   <services xmlns="http://ruggedcom.com/ns/rmf_services">
    <time>
     <ntp>
      <ntp-status/>
     </ntp>
    </time>
   </services>
  </data>
 </action>
</rpc>]]>]]>
```

Section 6.3.2
log (Link-Failover)

This action displays the link failover log. This action does not take any parameters.

```
<rpc message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <action xmlns="http://tail-f.com/ns/netconf/actions/1.0">
  <data>
   <services xmlns="http://ruggedcom.com/ns/rmf_services">
    <link-failover>
     <main></main>
     <log/>
    </link-failover>
   </services>
  </data>
 </action>
</rpc>]]>]]>
```

Section 6.3.3
start-test (Link Failover)

This action starts a test of the link failover function. Specify the name of the interface to test in the `<name>` element. Parameters include `<test-duration>` and `<start-test-delay>`.

```
<rpc message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <action xmlns="http://tail-f.com/ns/netconf/actions/1.0">
  <data>
```

```
   <services xmlns="http://ruggedcom.com/ns/rmf_services">
    <link-failover>
     <main>{interfaceName}</main>
     <start-test>
      <test-duration>{testDuration}</test-duration>
      <start-test-delay>{testDelay}</start-test-delay>
     </start-test>
    </link-failover>
   </services>
  </data>
 </action>
</rpc>]]>]]>
```

- `{interfaceName}`

  The name of the interface on which to perform the link failover test.

- `{testDuration}`

  The amount of time, in minutes, to run the test before restoring service to the main trunk.

- `{crlFileName}`

  The amount of time, in minutes, to wait before starting the link failover test.

Section 6.3.4
cancel-test (Link Failover)

This action stops the link failover test on the specified interface. Specify the name of the interface in the `<name>` element. This action does not take any parameters.

```
<rpc message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <action xmlns="http://tail-f.com/ns/netconf/actions/1.0">
  <data>
   <services xmlns="http://ruggedcom.com/ns/rmf_services">
    <link-failover>
     <main>{interfaceName}</main>
     <cancel-test/>
    </link-failover>
   </services>
  </data>
 </action>
</rpc>]]>]]>
```

- `{interfaceName}`

  The name of the interface on which to stop the link failover test.

Section 6.3.5
show-active-leases (DHCP Server)

This action returns a list of active leases from the DHCP server. This action does not take any parameters.

```
<rpc message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <action xmlns="http://tail-f.com/ns/netconf/actions/1.0">
  <data>
   <services xmlns="http://ruggedcom.com/ns/rmf_services">
```

```
    <dhcpserver>
     <show-active-leases/>
    </dhcpserver>
   </services>
  </data>
 </action>
</rpc>]]>]]>
```

Section 6.4

Switch Namespace Actions

Section 6.4.1

clear-stp-stats (Switch)

This action clears the spanning-tree protocol statistics. This action does not take any parameters.

```
<rpc message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <action xmlns="http://tail-f.com/ns/netconf/actions/1.0">
  <data>
   <switch xmlns="http://ruggedcom.com/ns/rmf_ifswitch">
    <spanning-tree>
     <clear-stp-stats/>
    </spanning-tree>
   </switch>
  </data>
 </action>
</rpc>]]>]]>
```

Section 6.4.2

flush-dynamic-rules (Switch)

This action deletes all dynamic entries from the routing-rules-summary table. This action does not take any parameters.

```
<rpc message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <action xmlns="http://tail-f.com/ns/netconf/actions/1.0">
  <data>
   <switch xmlns="http://ruggedcom.com/ns/rmf_ifswitch">
    <layer3-switching>
     <flush-dynamic-rules/>
    </layer3-switching>
   </switch>
  </data>
 </action>
</rpc>]]>]]>
```

Section 6.4.3
reset-all-switch-ports (Switch)

This action resets all switch ports. This action does not take any parameters.

```
<rpc message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <action xmlns="http://tail-f.com/ns/netconf/actions/1.0">
  <data>
   <switch xmlns="http://ruggedcom.com/ns/rmf_ifswitch">
    <reset-all-switch-ports/>
   </switch>
  </data>
 </action>
</rpc>]]>]]>
```

Section 6.4.4
clear-all-switch-stats (Switch)

This action clears all switch statistics. This action does not take any parameters.

```
<rpc message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <action xmlns="http://tail-f.com/ns/netconf/actions/1.0">
  <data>
   <switch xmlns="http://ruggedcom.com/ns/rmf_ifswitch">
    <clear-all-switch-stats/>
   </switch>
  </data>
 </action>
</rpc>]]>]]>
```

Section 6.4.5
clear-cable-stats-all (Switch)

This action clears all cable test diagnostic statistics. This action does not take any parameters.

```
<rpc message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <action xmlns="http://tail-f.com/ns/netconf/actions/1.0">
  <data>
   <switch xmlns="http://ruggedcom.com/ns/rmf_ifswitch">
    <clear-cable-stats-all/>
   </switch>
  </data>
 </action>
</rpc>]]>]]>
```

7 NETCONF Settings, Logs, and Statistics

This section describes the NETCONF settings, logs, and statistics reporting available in ROXII. This section describes how to use these features through the ROXII web interface.

Section 7.1
NETCONF Settings

On the NETCONF settings form, you configure the basic NETCONF subsystem parameters, including:

- enabling or disabling the NETCONF subsystem
- the default IP address on which the system listens for NETCONF activity
- the default port on which the system listens for NETCONF activity
- optional IP addresses and ports on which the system listens for NETCONF activity
- the maximum number of concurrent NETCONF sessions
- the idle timeout period for NETCONF sessions

To configure the NETCONF Subsystem, do the following:

1. Log in to the ROXII web management interface and enter an Edit mode.
2. Navigate to `</admin/netconf>`. The **NETCONF Sessions** form appears:

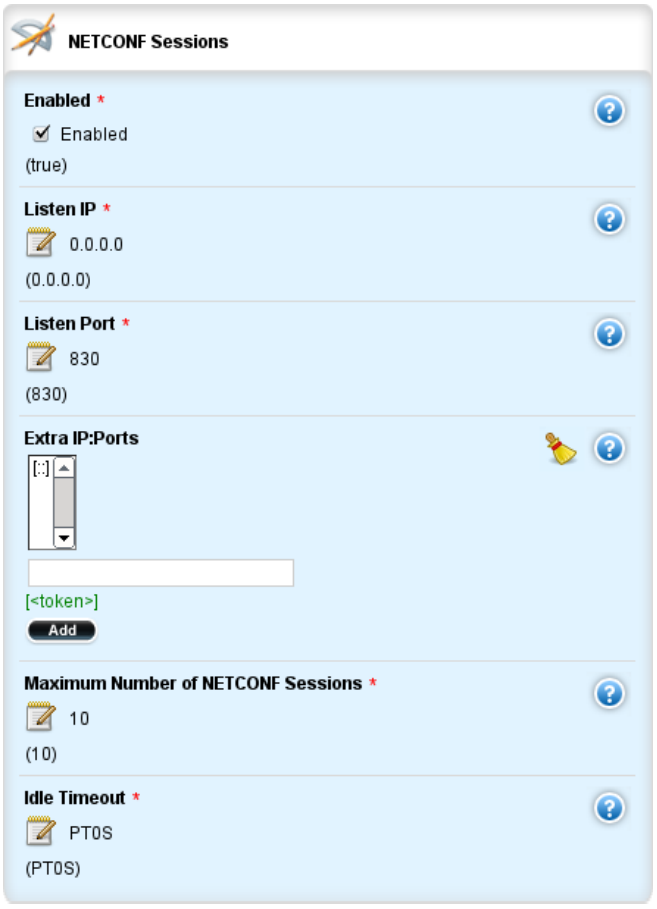

Figure 5: NETCONF Sessions form

3. Set the following parameters:

Parameter	Description
enabled enabled	**Synopsis:** true or false **Default:** true Provides the ability to configure NETCONF features on the device.
Listen IP listen-ip { listen-ip }	**Synopsis:** A string **Default:** 0.0.0.0 The IP Address the CLI will listen on for NETCONF requests.
Listen Port port { port }	**Synopsis:** An integer between 0 and 65535 **Default:** 830 The port on which NETCONF listens for NETCONF requests.
Extra IP:Ports extra-ip-ports { extra-ip-ports }	**Synopsis:** A string Additional IP addresses and ports on which NETCONF listens for NETCONF requests. You can specify IP addresses and ports in the following forms: <itemizedlist><listitem>nnn.nnn.nnn.nnn:port represents an IPv4 address followed by a colon and port number. For example, 192.168.10.12:19343</listitem> <listitem>0.0.0.0 represents the default IPv4 address and default port number. This is the default configuration.</listitem> <listitem>[::]:port represents an IPv6 address followed by a colon and port number. For example, [fe80::5eff:35ff]:16000</listitem> <listitem>If using

Parameter	Description
	the default address, do not specify another listen address with the same port.</listitem> </itemizedlist>
Maximum Number of NETCONF Sessions max-sessions { max-sessions }	**Synopsis:** { unbounded } **Default:** 10 The maximum number of concurrent NETCONF sessions.
Idle Timeout idle-timeout { idle-timeout }	**Synopsis:** A string **Default:** PT0S The maximum idle time before terminating a NETCONF session. If the session is waiting for notifications, or has a pending confirmed commit, the idle timeout is not used. A value of 0 means no timeout.
In Bad Hellos in-bad-hellos	The total number of sessions silently dropped because an invalid 'hello' message was received. This includes hello messages with a 'session-id' attribute, bad namespace, and bad capability declarations.
In Sessions in-sessions	The total number of NETCONF sessions started towards the NETCONF peer. inSessions - inBadHellos = 'The number of correctly started NETCONF sessions.'
Dropped Sessions dropped-sessions	The total number of NETCONF sessions dropped. inSessions - inBadHellos = 'The number of correctly started NETCONF sessions.'
In RPCs in-rpcs	The total number of RPC requests received.
In Bad RPCs in-bad-rpcs	The total number of RPCs which were parsed correctly, but couldn't be serviced because they contained non-conformant XML.
Out RPC Errors out-rpc-errors	The total number of 'rpc-reply' messages with 'rpc-error' sent.
Out Notifications out-notifications	The total number of 'notification' messages sent.

4. Commit the changes.

Section 7.2
NETCONF Logs

The system logs NETCONF activity in two log files:

- The NETCONF Summary Log records each NETCONF transaction. Each record includes the date and time of the transaction, the NETCONF session identifier, and a description of the transaction. For example:

```
.
.
.
<INFO> 5-Apr-2012::04:26:33.877 ruggedcom confd[2098]: netconf id=9450 new ssh session for user
 "admin" from 192.168.0.10
<INFO> 5-Apr-2012::04:27:03.574 ruggedcom confd[2098]: netconf id=9450 got rpc:
 {urn:ietf:params:xml:ns:netconf:base:1.0}validate attrs: message-id="103"
<INFO> 5-Apr-2012::04:27:04.167 ruggedcom confd[2098]: netconf id=9450 validate source=candidate
 attrs: message-id="103"
<INFO> 5-Apr-2012::04:27:06.691 ruggedcom confd[2098]: netconf id=9450 sending rpc-reply, attrs:
 message-id="103"
.
```

```
.
.
```

- The NETCONF Trace Log records the text of each NETCONF XML message received by and sent from the device. Each entry includes the NETCONF session identifier, whether the message is read (received by the device) or write (sent from the device), and the full text of the XML message. For example:

```
.
.
.
**> sess:9450 read:
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="103">
  <validate>
    <source>
      <running/>
    </source>
  </validate>
</rpc>

**< sess:9450 write:
<rpc-reply message-id="103" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <ok/>
</rpc-reply>
.
.
.
```

To enable or disable the NETCONF logs, do the following:

1. Log in to the ROXII web management interface and enter an Edit mode.
2. Navigate to `</admin/logging/diagnostics>`. The **NETCONF Summary Log** and **NETCONF Trace Log** forms appear:

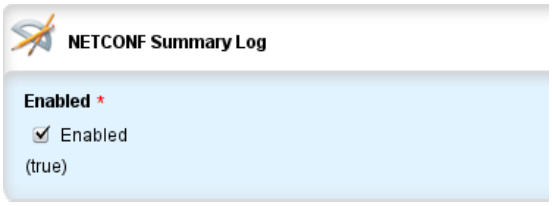

Figure 6: NETCONF Summary Log form

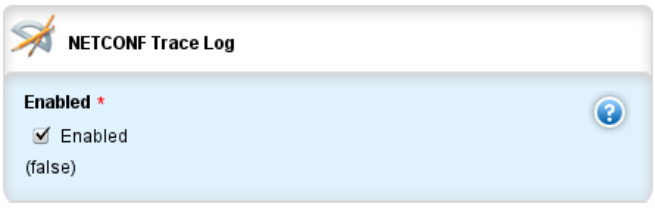

Figure 7: NETCONF Trace Log form

3. To enable a log, select the **Enabled** checkbox. To disable a log, clear the **Enabled** checkbox.
4. Commit the changes.

Section 7.3
NETCONF Statistics

The system reports on the following NETCONF statistics:

- the number of bad <hello> statements received by the device
- the number of incoming NETCONF sessions
- the number of dropped NETCONF sessions
- the number of incoming <rpc> requests
- the number of unparseable incoming <rpc> requests
- the number of outgoing <rpc-reply> messages with <rpc-error> indications
- the number of outgoing notification messages

To view NETCONF statistics in the ROXII web management interface, do the following:

1. Log in to the ROXII web management interface.
2. Navigate to </admin/netconf>. The **NETCONF State/Statistics** form appears:

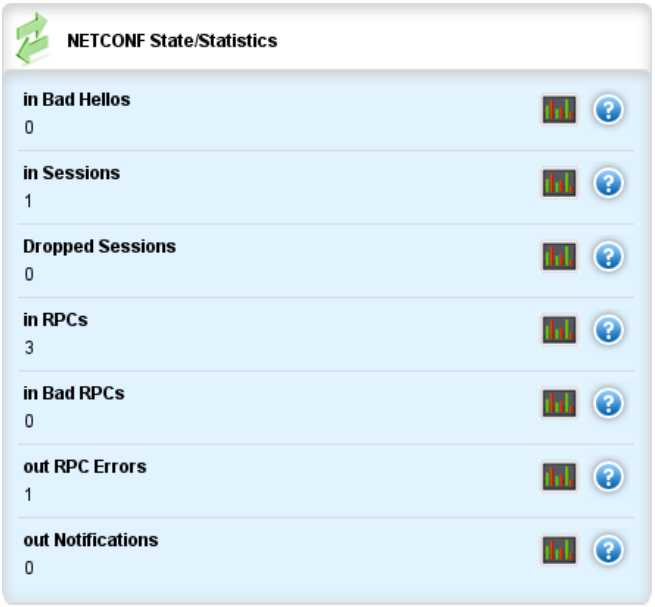

Figure 8: NETCONF State/Statistics form

3. Review the following information:

Parameter	Description
In Bad Hellos in-bad-hellos	The total number of sessions silently dropped because an invalid 'hello' message was received. This includes hello messages with a 'session-id' attribute, bad namespace, and bad capability declarations.
In Sessions in-sessions	The total number of NETCONF sessions started towards the NETCONF peer.

Parameter	Description
	inSessions - inBadHellos = 'The number of correctly started NETCONF sessions.'
Dropped Sessions dropped-sessions	The total number of NETCONF sessions dropped. inSessions - inBadHellos = 'The number of correctly started NETCONF sessions.'
In RPCs in-rpcs	The total number of RPC requests received.
In Bad RPCs in-bad-rpcs	The total number of RPCs which were parsed correctly, but couldn't be serviced because they contained non-conformant XML.
Out RPC Errors out-rpc-errors	The total number of 'rpc-reply' messages with 'rpc-error' sent.
Out Notifications out-notifications	The total number of 'notification' messages sent.

8 Examples

This section provides examples of how to set and retrieve data on a device. Each example shows how to set or retrieve a specific element or set of elements. Each example also demonstrates a general NETCONF concept, such as how to retrieve data from the running configuration, or how to use special attributes to delete or replace data.

Each example assumes that you are connected to a device and have established a NETCONF session. For instructions on how to establish a NETCONF session, seeChapter 3, *NETCONF Sessions*.

Each example provides all of the `<rpc>` commands necessary to perform the function.

Table: NETCONF Examples

NETCONF Example	Example demonstrates these techniques:
Section 8.1, “Getting the System Name”	Querying for running configuration data.
Section 8.2, “Getting the ROX Release”	Querying for fixed system data.
Section 8.3, “Getting the Chassis Status”	Querying for state information.
Section 8.4, “Setting the System Clock”	Using an action command.
Section 8.5, “Acknowledging Alarms”	Using an action command.
Section 8.6, “Clearing All Alarms”	Using an action command.
Section 8.7, “Viewing Alarms”	Querying for state information.
Section 8.8, “Restoring Factory Defaults”	Using an action command.
Section 8.9, “Changing the System Name by Locking and Committing”	Recommended editing procedure.
Section 8.10, “Changing the System Name Directly”	Editing configuration data in the running configuration.
Section 8.11, “Creating a Static VLAN”	Recommended editing procedure.
Section 8.12, “Assigning a PVID on a Port”	Recommended editing procedure.
Section 8.13, “Disabling Spanning Tree on a Specific Port”	Recommended editing procedure.
Section 8.14, “Configuring an IP Address on a Specific Port”	Recommended editing procedure.
Section 8.15, “Deleting an IP Address”	Recommended editing procedure.
Section 8.16, “Setting a Static Route”	Recommended editing procedure.
Section 8.17, “Disabling Spanning Tree Globally”	Recommended editing procedure. Deleting data with the**nc:operation="delete"** attribute.
Section 8.18, “Configuring WAN Interfaces”	Recommended editing procedure.
Section 8.19, “Configuring an IP Address on a WAN Interface”	Recommended editing procedure.
Section 8.20, “Enabling a WAN Port”	Recommended editing procedure.
Section 8.21, “Retrieving all IP Addresses from the Running Configuration”	Recommended editing procedure.
Section 8.22, “Retrieving the Active Routes on a Device”	Querying for running configuration data.

NETCONF Example	Example demonstrates these techniques:
Section 8.23, “Configuring Static Multicast Routing on a Layer 3 Device”	Recommended editing procedure.
Section 8.24, “Enabling Static Multicast Routing on a Layer 3 Device”	Recommended editing procedure.
Section 8.25, “Retrieving Static Multicast Status on a Layer 3 Device”	Querying for running configuration data.
Section 8.26, “Replacing an IP Address”	Recommended editing procedure. Replacing data with the**nc:operation="replace"** attribute.
Section 8.27, “Configuring a Port to Dynamically Obtain an IP Address”	Recommended editing procedure.
Section 8.28, “Configuring OSPF Area and Network on a Layer 3 Device”	Recommended editing procedure.
Section 8.29, “Enabling the OSPF Passive-Default Option”	Recommended editing procedure.
Section 8.30, “Configure an OSPF Non-passive Port”	Recommended editing procedure. Deleting information with the**nc:operation="delete"** attribute.
Section 8.31, “Configuring OSPF Parameters”	Recommended editing procedure.
Section 8.32, “Enabling the OSPF redistribute-connected Option”	Recommended editing procedure.
Section 8.33, “Enabling OSPF on a Layer 3 Device”	Recommended editing procedure.
Section 8.34, “Retrieving OSPF Status”	Querying for running configuration data.
Section 8.35, “Retrieving All Data from the Routing Namespace”	Querying for running configuration data.
Section 8.36, “Configuring DHCP Server”	Recommended editing procedure.
Section 8.37, “Configure the DHCP Server Port Listening for DHCP Client Requests”	Recommended editing procedure.
Section 8.38, “Enabling the DHCP Server Service”	Recommended editing procedure.
Section 8.39, “Disabling an Ethernet Port”	Recommended editing procedure. Deleting data with the**nc:operation="delete"** attribute.
Section 8.40, “Enabling an Ethernet Port”	Recommended editing procedure.
Section 8.41, “Checking an IP Address on a Specific Port using the Interfaces Namespace”	Querying for running configuration data.
Section 8.42, “Configuring Frame Relay on a WAN Port”	Recommended editing procedure.
Section 8.43, “Retreiving All Data From Running Database Including Default Values”	Querying for configuration data (including values) from a running database.
Section 8.44, “Retreiving All Data From Running Database Including Default Tags and Values”	Querying for configuration data (including tags and values) from a running database.
Section 8.45, “Changing a User's Password”	Querying for configuration data (including tags and values) from a running database.
Section 8.46, “Displaying the Status”	Displaying the status of the running service.
Section 8.47, “Installing a Certificate to an IPSec Connection”	Installing a certificate to the device.
Section 8.48, “Installing a CA Certificate”	Installing a CA certificate to the device.
Section 8.49, “Configuring a Signed CA Certificate”	Configuring a signed CA certificate.

NETCONF Example	Example demonstrates these techniques:
Section 8.50, “Installing a Private Key to a Signed CA Certificate”	Installing a private key to a signed CA certificate.
Section 8.51, “Installing a CRL File”	Installing a CRL file to the device.
Section 8.52, “Removing a Certificate”	Removing a certificate from the device.
Section 8.53, “Removing a CA certificate”	Removing a CA certificate from the device.
Section 8.54, “Removing a CRL File”	Removing a CRL file from the device.

Section 8.1
Getting the System Name

In this example, a single <rpc> queries the running configuration and returns the system name.

This example shows how to issue a query for configuration data directly from the running configuration.

```
<rpc message-id="2"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" with-defaults="true">
 <get-config>
  <source>
   <running/>
  </source>
  <filter type="subtree">
   <admin xmlns="http://ruggedcom.com/ns/rmf_admin">
    <system-name></system-name>
   </admin>
  </filter>
 </get-config>
</rpc>]]>]]>
```

Section 8.2
Getting the ROX Release

In this example, a single <rpc> queries the running configuration and returns the ROX release information.

This example shows how to issue a query for fixed system data from the device.

```
<rpc message-id="2"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <get>
  <filter type="subtree">
   <chassis xmlns="http://ruggedcom.com/ns/rmf_chassis">
    <chassis-status>
     <rox-release></rox-release>
    </chassis-status>
   </chassis>
  </filter>
 </get>
</rpc>]]>]]>
```

Section 8.3
Getting the Chassis Status

In this example, a single <rpc> queries retrieves the chassis status information from the device.

This example shows how to issue a query for state information directly from the device.

```
<rpc message-id="2"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <get>
  <filter type="subtree">
   <chassis xmlns="http://ruggedcom.com/ns/rmf_chassis">
    <status></status>
   </chassis>
  </filter>
 </get>
</rpc>]]>]]>
```

Section 8.4
Setting the System Clock

In this example, a single <rpc> sets the system clock with the **set-system-clock** action.

This example shows how to use a ROXII action on a running device.

```
<rpc message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <action xmlns="http://tail-f.com/ns/netconf/actions/1.0">
  <data>
   <admin xmlns="http://ruggedcom.com/ns/rmf_admin">
    <set-system-clock>
     <time>2011-01-12 17:52:25</time>
    </set-system-clock>
   </admin>
  </data>
 </action>
</rpc>]]>]]>
```

Section 8.5
Acknowledging Alarms

In this example, a single <rpc> acknowledges all alarms on the device with the **acknowledge-all-alarms** action.

This example shows how to use a ROXII action on a running device.

```
<rpc message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <action xmlns="http://tail-f.com/ns/netconf/actions/1.0">
  <data>
   <admin xmlns="http://ruggedcom.com/ns/rmf_admin">
    <acknowledge-all-alarms/>
   </admin>
  </data>
 </action>
```

```
</rpc>]]>]]>
```

Section 8.6
Clearing All Alarms

In this example, a single <rpc> clears all alarms on the device with the **clear-all-alarms** action.

This example shows how to use a ROXII action on a running device.

```
<rpc message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <action xmlns="http://tail-f.com/ns/netconf/actions/1.0">
  <data>
   <admin xmlns="http://ruggedcom.com/ns/rmf_admin">
    <clear-all-alarms/>
   </admin>
  </data>
 </action>
</rpc>]]>]]>
```

Section 8.7
Viewing Alarms

In this example, a single <rpc> queries the device and returns a list of active alarms.

This example shows how to issue a query for state information directly from the device.

```
<rpc message-id="2"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <get>
  <filter type="subtree">
   <admin xmlns="http://ruggedcom.com/ns/rmf_admin">
    <alarms></alarms>
   </admin>
  </filter>
 </get>
</rpc>]]>]]>
```

Section 8.8
Restoring Factory Defaults

In this example, a single <rpc> restores the factory default settings with the **restore-factory-defaults** action.

This example shows how to use a ROXII action on a running device.

```
<rpc message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <action xmlns="http://tail-f.com/ns/netconf/actions/1.0">
  <data>
   <admin xmlns="http://ruggedcom.com/ns/rmf_admin">
    <restore-factory-defaults/>
   </admin>
  </data>
 </action>
```

```
</rpc>]]>]]>
```

Section 8.9

Changing the System Name by Locking and Committing

In this example, multiple <rpc> requests change the system name in the candidate configuration and then commit the changes.

This example shows the recommended procedure for making configuration changes on a device.

Procedure: Changing the System Name

1. Discard uncommitted changes:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="232">
 <discard-changes/>
</rpc>]]>]]>
```

2. Lock the running configuration:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="230">
 <lock>
  <target>
   <running/>
  </target>
 </lock>
</rpc>]]>]]>
```

3. Lock the candidate configuration:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="231">
 <lock>
  <target>
   <candidate/>
  </target>
 </lock>
</rpc>]]>]]>
```

4. Change the system-name setting:

```
<rpc message-id="233"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <edit-config>
  <target>
   <candidate/>
  </target>
  <config>
   <admin xmlns="http://ruggedcom.com/ns/rmf_admin">
    <system-name>Substation 24 Locker 2 Router 1</system-name>
   </admin>
  </config>
 </edit-config>
</rpc>]]>]]>
```

5. Commit the changes:

```
<rpc message-id="234"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <commit/>
```

```
</rpc>]]>]]>
```

6. Unlock the candidate configuration:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="235">
 <unlock>
  <target>
   <candidate/>
  </target>
 </unlock>
</rpc>]]>]]>
```

7. Unlock the running configuration:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="236">
 <unlock>
  <target>
   <running/>
  </target>
 </unlock>
</rpc>]]>]]>
```

Section 8.10

Changing the System Name Directly

In this example, a single <rpc> request changes the system name directly in the running configuration.

This example shows how to change configuration data on the running configuration directly without locking the datastores. Changes made in this manner are applied to the running configuration immediately.

CAUTION!
Exercise caution when making changes directly to the running configuration. Making an error in the configuration settings may interrupt service.

```
<rpc message-id="233"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <edit-config>
  <target>
   <running/>
  </target>
  <config>
   <admin xmlns="http://ruggedcom.com/ns/rmf_admin">
    <system-name>REAL-Test</system-name>
   </admin>
  </config>
 </edit-config>
</rpc>]]>]]>
```

Section 8.11

Creating a Static VLAN

In this example, multiple <rpc> requests create a static VLAN in the candidate configuration and then commit the changes.

This example shows the recommended procedure for making configuration changes on a device.

Procedure: Creating a Static VLAN

1. Lock the running configuration:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="230">
 <lock>
  <target>
   <running/>
  </target>
 </lock>
</rpc>]]>]]>
```

2. Lock the candidate configuration:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="231">
 <lock>
  <target>
   <candidate/>
  </target>
 </lock>
</rpc>]]>]]>
```

3. Discard uncommitted changes:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="232">
 <discard-changes/>
</rpc>]]>]]>
```

4. Configure the static VLAN parameters:

```
<rpc message-id="233"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <edit-config>
  <target>
   <candidate/>
  </target>
  <config>
   <switch xmlns="http://ruggedcom.com/ns/rmf_ifswitch">
    <vlans>
     <static-vlan>
      <vid>0086</vid>
     </static-vlan>
    </vlans>
   </switch>
  </config>
 </edit-config>
</rpc>]]>]]>
```

5. Commit the changes:

```
<rpc message-id="234"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <commit/>
</rpc>]]>]]>
```

6. Unlock the candidate configuration:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="235">
 <unlock>
  <target>
   <candidate/>
  </target>
 </unlock>
</rpc>]]>]]>
```

7. Unlock the target configuration:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="236">
 <unlock>
  <target>
   <running/>
  </target>
 </unlock>
</rpc>]]>]]>
```

Section 8.12

Assigning a PVID on a Port

In this example, multiple <rpc> requests assign a PVID to a port in the candidate configuration and then commit the changes.

This example shows the recommended procedure for making configuration changes on a device.

Procedure: Assigning a PVID on a Port

1. Lock the running configuration:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="230">
 <lock>
  <target>
   <running/>
  </target>
 </lock>
</rpc>]]>]]>
```

2. Lock the candidate configuration:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="231">
 <lock>
  <target>
   <candidate/>
  </target>
 </lock>
</rpc>]]>]]>
```

3. Discard uncommitted changes:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="232">
 <discard-changes/>
</rpc>]]>]]>
```

4. Configure the PVID parameters:

```
<rpc message-id="233"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <edit-config>
  <target>
   <candidate/>
  </target>
  <config>
   <interface xmlns="http://ruggedcom.com/ns/rmf_if">
    <switch>
     <slot>lm4</slot>
     <port>6</port>
     <vlan>
      <pvid>0086</pvid>
      <format>untagged</format>
     </vlan>
```

```
    </switch>
   </interface>
  </config>
 </edit-config>
</rpc>]]>]]>
```

5. Commit the changes:

```
<rpc message-id="234"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <commit/>
</rpc>]]>]]>
```

6. Unlock the candidate configuration:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="235">
 <unlock>
  <target>
   <candidate/>
  </target>
 </unlock>
</rpc>]]>]]>
```

7. Unlock the running configuration:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="236">
 <unlock>
  <target>
   <running/>
  </target>
 </unlock>
</rpc>]]>]]>
```

Section 8.13

Disabling Spanning Tree on a Specific Port

In this example, multiple <rpc> requests disable spanning tree on a specified port in the candidate configuration and then commit the changes.

This example shows the recommended procedure for making configuration changes on a device.

Procedure: Disabling Spanning Tree on a Specific Port

1. Lock the running configuration:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="230">
 <lock>
  <target>
   <running/>
  </target>
 </lock>
</rpc>]]>]]>
```

2. Lock the candidate configuration:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="231">
 <lock>
  <target>
   <candidate/>
  </target>
 </lock>
</rpc>]]>]]>
```

3. Discard uncommitted changes:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="232">
 <discard-changes/>
</rpc>]]>]]>
```

4. Disable spanning tree with the**nc:delete** attribute:

```
<rpc message-id="233"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
<edit-config>
  <target>
   <candidate/>
  </target>
  <config>
   <interface xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0" xmlns="http://ruggedcom.com/ns/
rmf_if">
    <switch>
     <slot>lm4</slot>
     <port>6</port>
     <spanning-tree>
      <enabled nc:operation="delete"/>
     </spanning-tree>
    </switch>
   </interface>
  </config>
 </edit-config>
</rpc>]]>]]>
```

5. Commit the changes:

```
<rpc message-id="234"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <commit/>
</rpc>]]>]]>
```

6. Unlock the candidate configuration:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="235">
 <unlock>
  <target>
   <candidate/>
  </target>
 </unlock>
</rpc>]]>]]>
```

7. Unlock the running configuration:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="236">
 <unlock>
  <target>
   <running/>
  </target>
 </unlock>
</rpc>]]>]]>
```

Section 8.14

Configuring an IP Address on a Specific Port

In this example, multiple <rpc> requests configure an IP address on a specified port in the candidate configuration and then commit the changes.

This example shows the recommended procedure for making configuration changes on a device.

Procedure: Configuring an IP Address on a Specific Port

1. Lock the running configuration:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="230">
 <lock>
  <target>
   <running/>
  </target>
 </lock>
</rpc>]]>]]>
```

2. Lock the candidate configuration:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="231">
 <lock>
  <target>
   <candidate/>
  </target>
 </lock>
</rpc>]]>]]>
```

3. Discard uncommitted changes:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="232">
 <discard-changes/>
</rpc>]]>]]>
```

4. Set the IP address on the port:

```
<rpc message-id="233"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <edit-config>
  <target>
   <candidate/>
  </target>
  <config>
   <ip xmlns="http://ruggedcom.com/ns/rmf_ip">
    <ifname>fe-cm-1</ifname>
    <ipv4>
     <address>
      <ipaddress>192.168.1.43/24</ipaddress>
     </address>
    </ipv4>
   </ip>
  </config>
 </edit-config>
</rpc>]]>]]>
```

5. Commit the changes:

```
<rpc message-id="234"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <commit/>
</rpc>]]>]]>
```

6. Unlock the candidate configuration:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="235">
 <unlock>
  <target>
   <candidate/>
  </target>
 </unlock>
```

```
</rpc>]]>]]>
```

7. Unlock the running configuration:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="236">
 <unlock>
  <target>
   <running/>
  </target>
 </unlock>
</rpc>]]>]]>
```

Section 8.15

Deleting an IP Address

In this example, multiple <rpc> requests delete an IP address in the candidate configuration and then commit the changes.

This example shows the recommended procedure for making configuration changes on a device.

This example also shows how container elements need to be deleted for some elements. In this example, the actual IP address is in the <ipaddress> element, which is within an<address> container element. To properly delete an IP address, you must delete both the element holding the address and its container element.

Procedure: Deleting an IP Address

1. Lock the running configuration:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="230">
 <lock>
  <target>
   <running/>
  </target>
 </lock>
</rpc>]]>]]>
```

2. Lock the candidate configuration:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="231">
 <lock>
  <target>
   <candidate/>
  </target>
 </lock>
</rpc>]]>]]>
```

3. Discard uncommitted changes:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="232">
 <discard-changes/>
</rpc>]]>]]>
```

4. Delete the IP address and its container element:

```
<rpc message-id="233"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <edit-config>
  <target>
   <candidate/>
  </target>
  <config>
   <ip xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0" xmlns="http://ruggedcom.com/ns/rmf_ip">
```

```
    <ifname>fe-cm-1</ifname>
    <ipv4>
     <address nc:operation="delete">
      <ipaddress nc:operation="delete">192.168.1.2/24</ipaddress>
     </address>
    </ipv4>
   </ip>
  </config>
 </edit-config>
</rpc>]]>]]>
```

5. Commit the changes:

```
<rpc message-id="234"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <commit/>
</rpc>]]>]]>
```

6. Unlock the candidate configuration:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="235">
 <unlock>
  <target>
   <candidate/>
  </target>
 </unlock>
</rpc>]]>]]>
```

7. Unlock the running configuration:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="236">
 <unlock>
  <target>
   <running/>
  </target>
 </unlock>
</rpc>]]>]]>
```

Section 8.16

Setting a Static Route

In this example, multiple `<rpc>` requests set a static route in the candidate configuration and then commit the changes.

This example shows the recommended procedure for making configuration changes on a device.

Procedure: Setting a Static Route

1. Lock the running configuration:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="230">
 <lock>
  <target>
   <running/>
  </target>
 </lock>
</rpc>]]>]]>
```

2. Lock the candidate configuration:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="231">
 <lock>
```

```
  <target>
   <candidate/>
  </target>
 </lock>
</rpc>]]>]]>
```

3. Discard uncommitted changes:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="232">
 <discard-changes/>
</rpc>]]>]]>
```

4. Define the static route:

```
<rpc message-id="233"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <edit-config>
  <target>
   <candidate/>
  </target>
  <config>
   <routing xmlns="http://ruggedcom.com/ns/rmf_routing">
    <static>
     <ipv4>
      <route>
       <network>10.200.16.0/20</network>
       <via>
        <gw>172.30.128.1</gw>
       </via>
      </route>
     </ipv4>
    </static>
   </routing>
  </config>
 </edit-config>
</rpc>]]>]]>
```

5. Commit the changes:

```
<rpc message-id="234"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <commit/>
</rpc>]]>]]>
```

6. Unlock the candidate configuration:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="235">
 <unlock>
  <target>
   <candidate/>
  </target>
 </unlock>
</rpc>]]>]]>
```

7. Unlock the running configuration:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="236">
 <unlock>
  <target>
   <running/>
  </target>
 </unlock>
</rpc>]]>]]>
```

Section 8.17
Disabling Spanning Tree Globally

In this example, multiple <rpc> requests globally disable spanning tree in the candidate configuration and then commit the changes.

This example shows the recommended procedure for making configuration changes on a device. This example also shows how to delete data with the**nc:operation="delete"** attribute.

Procedure: Disabling Spanning Tree Globally

1. Lock the running configuration:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="230">
 <lock>
  <target>
   <running/>
  </target>
 </lock>
</rpc>]]>]]>
```

2. Lock the candidate configuration:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="231">
 <lock>
  <target>
   <candidate/>
  </target>
 </lock>
</rpc>]]>]]>
```

3. Discard uncommitted changes:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="232">
 <discard-changes/>
</rpc>]]>]]>
```

4. Disable spanning tree:

```
<rpc message-id="233"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <edit-config>
  <target>
   <candidate/>
  </target>
  <config>
   <switch xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0" xmlns="http://ruggedcom.com/ns/
rmf_ifswitch">
    <spanning-tree>
     <enabled nc:operation="delete" />
    </spanning-tree>
   </switch>
  </config>
 </edit-config>
</rpc>]]>]]>
```

5. Commit the changes:

```
<rpc message-id="234"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <commit/>
</rpc>]]>]]>
```

6. Unlock the candidate configuration:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="235">
 <unlock>
  <target>
   <candidate/>
  </target>
 </unlock>
</rpc>]]>]]>
```

7. Unlock the running configuration:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="236">
 <unlock>
  <target>
   <running/>
  </target>
 </unlock>
</rpc>]]>]]>
```

Section 8.18

Configuring WAN Interfaces

In this example, multiple <rpc> requests configure a WAN interface in the candidate configuration and then commit the changes.

This example shows the recommended procedure for making configuration changes on a device.

Procedure: Configuring WAN Interfaces

1. Lock the running configuration:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="230">
 <lock>
  <target>
   <running/>
  </target>
 </lock>
</rpc>]]>]]>
```

2. Lock the candidate configuration:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="231">
 <lock>
  <target>
   <candidate/>
  </target>
 </lock>
</rpc>]]>]]>
```

3. Discard uncommitted changes:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="232">
 <discard-changes/>
</rpc>]]>]]>
```

4. Configure the WAN interface:

```
<rpc message-id="233"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <edit-config>
  <target>
   <candidate/>
```

```
  </target>
  <config>
   <interface xmlns="http://ruggedcom.com/ns/rmf_if">
    <wan>
     <slot>lm3</slot>
     <port>1</port>
     <t1>
      <t1params>
       <clock>master</clock>
      </t1params>
      <channel>
       <channelnumber>01</channelnumber>
       <ts>all</ts>
       <connection>
        <ppp>
         <nomagic>false</nomagic>
        </ppp>
       </connection>
      </channel>
     </t1>
    </wan>
   </interface>
  </config>
 </edit-config>
</rpc>]]>]]>
```

5. Commit the changes:

```
<rpc message-id="234"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <commit/>
</rpc>]]>]]>
```

6. Unlock the candidate configuration:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="235">
 <unlock>
  <target>
   <candidate/>
  </target>
 </unlock>
</rpc>]]>]]>
```

7. Unlock the running configuration:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="236">
 <unlock>
  <target>
   <running/>
  </target>
 </unlock>
</rpc>]]>]]>
```

Section 8.19

Configuring an IP Address on a WAN Interface

In this example, multiple `<rpc>` requests configure an IP address on a WAN interface in the candidate configuration and then commit the changes.

This example shows the recommended procedure for making configuration changes on a device.

Procedure: Configuring an IP Address on a WAN Interface

1. Lock the running configuration:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="230">
 <lock>
  <target>
   <running/>
  </target>
 </lock>
</rpc>]]>]]>
```

2. Lock the candidate configuration:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="231">
 <lock>
  <target>
   <candidate/>
  </target>
 </lock>
</rpc>]]>]]>
```

3. Discard uncommitted changes:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="232">
 <discard-changes/>
</rpc>]]>]]>
```

4. Configure the IP address on the WAN interface:

```
<rpc message-id="233"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <edit-config>
  <target>
   <candidate/>
  </target>
  <config>
   <ip xmlns="http://ruggedcom.com/ns/rmf_ip">
    <ifname>te1-3-1c01ppp</ifname>
    <ipv4>
     <address>
      <ipaddress>9.9.9.33/32</ipaddress>
      <peer>9.9.9.34</peer>
     </address>
    </ipv4>
   </ip>
  </config>
 </edit-config>
</rpc>]]>]]>
```

5. Commit the changes:

```
<rpc message-id="234"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <commit/>
</rpc>]]>]]>
```

6. Unlock the candidate configuration:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="235">
 <unlock>
  <target>
   <candidate/>
  </target>
 </unlock>
</rpc>]]>]]>
```

7. Unlock the running configuration:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="236">
 <unlock>
  <target>
   <running/>
  </target>
 </unlock>
</rpc>]]>]]>
```

Section 8.20

Enabling a WAN Port

In this example, multiple <rpc> requests enable a WAN port in the candidate configuration and then commit the changes.

This example shows the recommended procedure for making configuration changes on a device.

Procedure: Enabling a WAN Port

1. Lock the running configuration:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="230">
 <lock>
  <target>
   <running/>
  </target>
 </lock>
</rpc>]]>]]>
```

2. Lock the candidate configuration:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="231">
 <lock>
  <target>
   <candidate/>
  </target>
 </lock>
</rpc>]]>]]>
```

3. Discard uncommitted changes:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="232">
 <discard-changes/>
</rpc>]]>]]>
```

4. Enable the WAN port:

```
<rpc message-id="233"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <edit-config>
  <target>
   <candidate/>
  </target>
  <config>
   <interface xmlns="http://ruggedcom.com/ns/rmf_if">
    <wan>
     <slot>lm3</slot>
     <port>1</port>
     <enabled/>
    </wan>
   </interface>
```

```
  </config>
 </edit-config>
</rpc>]]>]]>
```

5. Commit the changes:

```
<rpc message-id="234"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <commit/>
</rpc>]]>]]>
```

6. Unlock the candidate configuration:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="235">
 <unlock>
  <target>
   <candidate/>
  </target>
 </unlock>
</rpc>]]>]]>
```

7. Unlock the running configuration:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="236">
 <unlock>
  <target>
   <running/>
  </target>
 </unlock>
</rpc>]]>]]>
```

Section 8.21

Retrieving all IP Addresses from the Running Configuration

In this example, a single `<rpc>` request retrieves all IP addresses from the running configuration on a device.

This example shows the typical procedure for querying data from the running configuration.

Procedure: Retrieving all IP Addresses from the Running Configuration

- Request the data from the running configuration:

```
<rpc message-id="2"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <get-config>
  <source>
   <running/>
  </source>
  <filter type="subtree">
   <ip xmlns="http://ruggedcom.com/ns/rmf_ip">
    <ipv4>
     <address></address>
    </ipv4>
   </ip>
  </filter>
 </get-config>
</rpc>]]>]]>
```

Section 8.22
Retrieving the Active Routes on a Device

In this example, a single <rpc> request retrieves the active routes from the running configuration on a device.

This example shows the typical procedure for querying data from the running configuration.

Procedure: Retrieving the Active Routes on a Device

- Request the data from the running configuration:

```
<rpc message-id="2"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <get>
  <filter type="subtree">
   <routing xmlns="http://ruggedcom.com/ns/rmf_routing">
    <status>
     <ipv4routes>
      <active-routes>
       <destination>default</destination>
       <gateway></gateway>
       <interface></interface>
       <type></type>
      </active-routes>
     </ipv4routes>
    </status>
   </routing>
  </filter>
 </get>
</rpc>]]>]]>
```

Section 8.23
Configuring Static Multicast Routing on a Layer 3 Device

In this example, multiple <rpc> requests enable static multicast routing in the candidate configuration and then commit the changes.

This example shows the recommended procedure for making configuration changes on a device.

Procedure: Configuring Static Multicast Routing on a Layer 3 Device

1. Lock the running configuration:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="230">
 <lock>
  <target>
   <running/>
  </target>
 </lock>
</rpc>]]>]]>
```

2. Lock the candidate configuration:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="231">
 <lock>
  <target>
   <candidate/>
  </target>
```

```
 </lock>
</rpc>]]>]]>
```

3. Discard uncommitted changes:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="232">
 <discard-changes/>
</rpc>]]>]]>
```

4. Configure static multicast routing:

```
<rpc message-id="233"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <edit-config>
  <target>
   <candidate/>
  </target>
  <config>
   <routing xmlns="http://ruggedcom.com/ns/rmf_routing">
    <multicast>
     <static>
      <mcast-groups>
       <description>Ruggedtest</description>
       <source-ip>192.168.1.155</source-ip>
       <multicast-ip>229.0.0.100</multicast-ip>
       <in-interface>fe-cm-1</in-interface>
       <if-change>-</if-change>
       <out-interface>
        <ifname>switch.0021</ifname>
       </out-interface>
      </mcast-groups>
     </static>
    </multicast>
   </routing>
  </config>
 </edit-config>
</rpc>]]>]]>
```

5. Commit the changes:

```
<rpc message-id="234"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <commit/>
</rpc>]]>]]>
```

6. Unlock the candidate configuration:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="235">
 <unlock>
  <target>
   <candidate/>
  </target>
 </unlock>
</rpc>]]>]]>
```

7. Unlock the running configuration:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="236">
 <unlock>
  <target>
   <running/>
  </target>
 </unlock>
</rpc>]]>]]>
```

Section 8.24

Enabling Static Multicast Routing on a Layer 3 Device

In this example, multiple <rpc> requests enable static multicast routing in the candidate configuration and then commit the changes.

This example shows the recommended procedure for making configuration changes on a device.

Procedure: Enabling Static Multicast Routing on a Layer 3 Device

1. Lock the running configuration:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="230">
 <lock>
  <target>
   <running/>
  </target>
 </lock>
</rpc>]]>]]>
```

2. Lock the candidate configuration:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="231">
 <lock>
  <target>
   <candidate/>
  </target>
 </lock>
</rpc>]]>]]>
```

3. Discard uncommitted changes:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="232">
 <discard-changes/>
</rpc>]]>]]>
```

4. Enable static multicast routing:

```
<rpc message-id="233"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <edit-config>
  <target>
   <candidate/>
  </target>
  <config>
   <routing xmlns="http://ruggedcom.com/ns/rmf_routing">
    <multicast>
     <static>
      <enabled/>
     </static>
    </multicast>
   </routing>
  </config>
 </edit-config>
</rpc>]]>]]>
```

5. Commit the changes:

```
<rpc message-id="234"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <commit/>
```

```
</rpc>]]>]]>
```

6. Unlock the candidate configuration:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="235">
 <unlock>
  <target>
   <candidate/>
  </target>
 </unlock>
</rpc>]]>]]>
```

7. Unlock the running configuration:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="236">
 <unlock>
  <target>
   <running/>
  </target>
 </unlock>
</rpc>]]>]]>
```

Section 8.25

Retrieving Static Multicast Status on a Layer 3 Device

In this example, a single <rpc> request retrieves the static multicast status information from the device.

This example shows the typical procedure for querying data from the running configuration.

Procedure: Retrieving Static Multicast Status on a Layer 3 Device

- Request the data from the running configuration:

```
<rpc message-id="2"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <get>
  <filter type="subtree">
   <routing xmlns="http://ruggedcom.com/ns/rmf_routing">
    <multicast>
     <static>
      <status></status>
     </static>
    </multicast>
   </routing>
  </filter>
 </get>
</rpc>]]>]]>
```

Section 8.26

Replacing an IP Address

In this example, multiple <rpc> requests replace an IP address in the candidate configuration and then commit the changes.

This example shows the recommended procedure for making configuration changes on a device. This example also shows how to use the**nc:operation="replace"** attribute to replace an existing value with a new value.

Procedure: Replacing an IP Address

1. Lock the running configuration:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="230">
 <lock>
  <target>
   <running/>
  </target>
 </lock>
</rpc>]]>]]>
```

2. Lock the candidate configuration:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="231">
 <lock>
  <target>
   <candidate/>
  </target>
 </lock>
</rpc>]]>]]>
```

3. Discard uncommitted changes:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="232">
 <discard-changes/>
</rpc>]]>]]>
```

4. Replace the IP address:

```
<rpc message-id="233"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <edit-config>
  <target>
   <candidate/>
  </target>
  <config>
   <ip xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0" xmlns="http://ruggedcom.com/ns/rmf_ip">
    <ifname>fe-4-2</ifname>
    <ipv4 nc:operation="replace">
     <address>
      <ipaddress>192.168.114.5/24</ipaddress>
     </address>
    </ipv4>
   </ip>
  </config>
 </edit-config>
</rpc>]]>]]>
```

5. Commit the changes:

```
<rpc message-id="234"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <commit/>
</rpc>]]>]]>
```

6. Unlock the candidate configuration:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="235">
 <unlock>
  <target>
   <candidate/>
  </target>
```

```
 </unlock>
</rpc>]]>]]>
```

7. Unlock the running configuration:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="236">
 <unlock>
  <target>
   <running/>
  </target>
 </unlock>
</rpc>]]>]]>
```

Section 8.27

Configuring a Port to Dynamically Obtain an IP Address

In this example, multiple <rpc> requests configure a report as a DHCP client in the candidate configuration and then commit the changes.

This example shows the recommended procedure for making configuration changes on a device.

Procedure: Configuring a Port to Dynamically Obtain an IP Address

1. Discard uncommitted changes:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="232">
 <discard-changes/>
</rpc>]]>]]>
```

2. Lock the running configuration:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="230">
 <lock>
  <target>
   <running/>
  </target>
 </lock>
</rpc>]]>]]>
```

3. Lock the candidate configuration:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="231">
 <lock>
  <target>
   <candidate/>
  </target>
 </lock>
</rpc>]]>]]>
```

4. Configure the <ip-address-src> setting for the port:

```
<rpc message-id="233"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <edit-config>
  <target>
   <candidate/>
  </target>
  <config>
   <interface xmlns="http://ruggedcom.com/ns/rmf_if">
```

```
    <eth>
     <slot>lm4</slot>
     <port>1</port>
     <ip-address-src>dynamic</ip-address-src>
    </eth>
   </interface>
  </config>
 </edit-config>
</rpc>]]>]]>
```

5. Commit the changes:

```
<rpc message-id="234"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <commit/>
</rpc>]]>]]>
```

6. Unlock the candidate configuration:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="235">
 <unlock>
  <target>
   <candidate/>
  </target>
 </unlock>
</rpc>]]>]]>
```

7. Unlock the running configuration:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="236">
 <unlock>
  <target>
   <running/>
  </target>
 </unlock>
</rpc>]]>]]>
```

Section 8.28

Configuring OSPF Area and Network on a Layer 3 Device

In this example, multiple <rpc> requests configure OSPF area and network in the candidate configuration and then commit the changes.

This example shows the recommended procedure for making configuration changes on a device.

Procedure: Configuring OSPF Area and Network on a Layer 3 Device

1. Lock the running configuration:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="230">
 <lock>
  <target>
   <running/>
  </target>
 </lock>
</rpc>]]>]]>
```

2. Lock the candidate configuration:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="231">
 <lock>
  <target>
   <candidate/>
  </target>
 </lock>
</rpc>]]>]]>
```

3. Discard uncommitted changes:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="232">
 <discard-changes/>
</rpc>]]>]]>
```

4. Configure the OSPF area and network:

```
<rpc message-id="233"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <edit-config>
  <target>
   <candidate/>
  </target>
  <config>
   <routing xmlns="http://ruggedcom.com/ns/rmf_routing">
    <dynamic>
     <ospf>
      <area>
       <area>0.0.0.0</area>
       <network>192.168.114.0/24</network>
      </area>
     </ospf>
    </dynamic>
   </routing>
  </config>
 </edit-config>
</rpc>]]>]]>
```

5. Commit the changes:

```
<rpc message-id="234"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <commit/>
</rpc>]]>]]>
```

6. Unlock the candidate configuration:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="235">
 <unlock>
  <target>
   <candidate/>
  </target>
 </unlock>
</rpc>]]>]]>
```

7. Unlock the running configuration:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="236">
 <unlock>
  <target>
   <running/>
  </target>
 </unlock>
</rpc>]]>]]>
```

Section 8.29

Enabling the OSPF Passive-Default Option

In this example, multiple <rpc> requests configure the OSPF passive-default option in the candidate configuration and then commit the changes.

This example shows the recommended procedure for making configuration changes on a device.

Procedure: Enabling the OSPF Passive-Default Option

1. Lock the running configuration:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="230">
 <lock>
  <target>
   <running/>
  </target>
 </lock>
</rpc>]]>]]>
```

2. Lock the candidate configuration:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="231">
 <lock>
  <target>
   <candidate/>
  </target>
 </lock>
</rpc>]]>]]>
```

3. Discard uncommitted changes:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="232">
 <discard-changes/>
</rpc>]]>]]>
```

4. Enable the OSPF<passive-default> option:

```
<rpc message-id="233"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <edit-config>
  <target>
   <candidate/>
  </target>
  <config>
   <routing xmlns="http://ruggedcom.com/ns/rmf_routing">
    <dynamic>
     <ospf>
      <passive-default/>
     </ospf>
    </dynamic>
   </routing>
  </config>
 </edit-config>
</rpc>]]>]]>
```

5. Commit the changes:

```
<rpc message-id="234"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <commit/>
</rpc>]]>]]>
```

6. Unlock the candidate configuration:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="235">
 <unlock>
  <target>
   <candidate/>
  </target>
 </unlock>
</rpc>]]>]]>
```

7. Unlock the running configuration:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="236">
 <unlock>
  <target>
   <running/>
  </target>
 </unlock>
</rpc>]]>]]>
```

Section 8.30

Configure an OSPF Non-passive Port

In this example, multiple <rpc> requests configure on OSPF port as non-passive in the candidate configuration and then commit the changes.

This example shows the recommended procedure for making configuration changes on a device. This example also shows how to use the**nc:operation="delete"** attribute to disable a configuration option.

Procedure: Configure an OSPF Non-passive Port

1. Lock the running configuration:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="230">
 <lock>
  <target>
   <running/>
  </target>
 </lock>
</rpc>]]>]]>
```

2. Lock the candidate configuration:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="231">
 <lock>
  <target>
   <candidate/>
  </target>
 </lock>
</rpc>]]>]]>
```

3. Discard uncommitted changes:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="232">
 <discard-changes/>
</rpc>]]>]]>
```

4. Delete the <passive> setting for the port:

```
<rpc message-id="233"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <edit-config>
  <target>
```

```
   <candidate/>
  </target>
  <config>
   <routing xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0" xmlns="http://ruggedcom.com/ns/
rmf_routing">
    <dynamic>
     <ospf>
      <interface>
       <ifname>switch.0022</ifname>
       <passive nc:operation="delete"/>
      </interface>
     </ospf>
    </dynamic>
   </routing>
  </config>
 </edit-config>
</rpc>]]>]]>
```

5. Commit the changes:

```
<rpc message-id="234"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <commit/>
</rpc>]]>]]>
```

6. Unlock the candidate configuration:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="235">
 <unlock>
  <target>
   <candidate/>
  </target>
 </unlock>
</rpc>]]>]]>
```

7. Unlock the running configuration:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="236">
 <unlock>
  <target>
   <running/>
  </target>
 </unlock>
</rpc>]]>]]>
```

Section 8.31

Configuring OSPF Parameters

In this example, multiple <rpc> requests configure OSPF parameters in the candidate configuration and then commit the changes.

This example shows the recommended procedure for making configuration changes on a device.

Procedure: Configuring OSPF Parameters

1. Lock the running configuration:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="230">
 <lock>
  <target>
   <running/>
  </target>
```

```
 </lock>
</rpc>]]>]]>
```

2. Lock the candidate configuration:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="231">
 <lock>
  <target>
   <candidate/>
  </target>
 </lock>
</rpc>]]>]]>
```

3. Discard uncommitted changes:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="232">
 <discard-changes/>
</rpc>]]>]]>
```

4. Configure the OSPF parameters:

```
<rpc message-id="233"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <edit-config>
  <target>
   <candidate/>
  </target>
  <config>
   <routing xmlns="http://ruggedcom.com/ns/rmf_routing">
    <dynamic>
     <ospf>
      <router-id>192.168.1.43</router-id>
      <interface>
       <ifname>switch.0022</ifname>
       <hello-interval>10</hello-interval>
       <dead-interval>
        <dead-interval>40</dead-interval>
       </dead-interval>
      </interface>
     </ospf>
    </dynamic>
   </routing>
  </config>
 </edit-config>
</rpc>]]>]]>
```

5. Commit the changes:

```
<rpc message-id="234"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <commit/>
</rpc>]]>]]>
```

6. Unlock the candidate configuration:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="235">
 <unlock>
  <target>
   <candidate/>
  </target>
 </unlock>
</rpc>]]>]]>
```

7. Unlock the running configuration:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="236">
```

```
 <unlock>
  <target>
   <running/>
  </target>
 </unlock>
</rpc>]]>]]>
```

Section 8.32

Enabling the OSPF redistribute-connected Option

In this example, multiple <rpc> requests enable the OSPF redistribute-connected option in the candidate configuration and then commit the changes.

This example shows the recommended procedure for making configuration changes on a device.

Procedure: Enabling the OSPF redistribute-connected Option

1. Lock the running configuration:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="230">
 <lock>
  <target>
   <running/>
  </target>
 </lock>
</rpc>]]>]]>
```

2. Lock the candidate configuration:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="231">
 <lock>
  <target>
   <candidate/>
  </target>
 </lock>
</rpc>]]>]]>
```

3. Discard uncommitted changes:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="232">
 <discard-changes/>
</rpc>]]>]]>
```

4. Enable the redistribute-connected option:

```
<rpc message-id="233"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <edit-config>
  <target>
   <candidate/>
  </target>
  <config>
   <routing xmlns="http://ruggedcom.com/ns/rmf_routing">
    <dynamic>
     <ospf>
      <redistribute>
       <type>connected</type>
      </redistribute>
     </ospf>
    </dynamic>
   </routing>
  </config>
```

```
 </edit-config>
</rpc>]]>]]>
```

5. Commit the changes:

```
<rpc message-id="234"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <commit/>
</rpc>]]>]]>
```

6. Unlock the candidate configuration:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="235">
 <unlock>
  <target>
   <candidate/>
  </target>
 </unlock>
</rpc>]]>]]>
```

7. Unlock the running configuration:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="236">
 <unlock>
  <target>
   <running/>
  </target>
 </unlock>
</rpc>]]>]]>
```

Section 8.33

Enabling OSPF on a Layer 3 Device

In this example, multiple `<rpc>` requests enable OSPF in the candidate configuration and then commit the changes.

This example shows the recommended procedure for making configuration changes on a device.

Procedure: Enabling OSPF on a Layer 3 Device

1. Lock the running configuration:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="230">
 <lock>
  <target>
   <running/>
  </target>
 </lock>
</rpc>]]>]]>
```

2. Lock the candidate configuration:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="231">
 <lock>
  <target>
   <candidate/>
  </target>
 </lock>
</rpc>]]>]]>
```

3. Discard uncommitted changes:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="232">
 <discard-changes/>
</rpc>]]>]]>
```

4. Enable OSPF:

```
<rpc message-id="233"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <edit-config>
  <target>
   <candidate/>
  </target>
  <config>
   <routing xmlns="http://ruggedcom.com/ns/rmf_routing">
    <dynamic>
     <ospf>
      <enabled/>
     </ospf>
    </dynamic>
   </routing>
  </config>
 </edit-config>
</rpc>]]>]]>
```

5. Commit the changes:

```
<rpc message-id="234"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <commit/>
</rpc>]]>]]>
```

6. Unlock the candidate configuration:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="235">
 <unlock>
  <target>
   <candidate/>
  </target>
 </unlock>
</rpc>]]>]]>
```

7. Unlock the running configuration:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="236">
 <unlock>
  <target>
   <running/>
  </target>
 </unlock>
</rpc>]]>]]>
```

Section 8.34

Retrieving OSPF Status

In this example, a single `<rpc>` request retrieves the OSPF status information from the running configuration on a device.

This example shows the typical procedure for querying data from the running configuration.

Procedure: Retrieving OSPF Status

- Request the data from the running configuration:

```
<rpc message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <action xmlns="http://tail-f.com/ns/netconf/actions/1.0">
  <data>
   <routing xmlns="http://ruggedcom.com/ns/rmf_routing">
    <status>
     <ospf_status/>
    </status>
   </routing>
  </data>
 </action>
</rpc>]]>]]>
```

Section 8.35

Retrieving All Data from the Routing Namespace

In this example, a single <rpc> request retrieves all configuration data from the Routing namespace.

This example shows the typical procedure for querying data from the running configuration.

Procedure: Retrieving All Data from the Routing Namespace

- Request the data from the running configuration:

```
<rpc message-id="2"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <get>
  <filter type="subtree">
   <routing xmlns="http://ruggedcom.com/ns/rmf_routing">
    <status></status>
   </routing>
  </filter>
 </get>
</rpc>]]>]]>
```

Section 8.36

Configuring DHCP Server

In this example, multiple <rpc> requests configure the DHCP server service in the candidate configuration and then commit the changes.

This example shows the recommended procedure for making configuration changes on a device.

Procedure: Configuring DHCP Server

1. Lock the running configuration:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="230">
 <lock>
  <target>
   <running/>
  </target>
 </lock>
</rpc>]]>]]>
```

2. Lock the candidate configuration:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="231">
 <lock>
  <target>
   <candidate/>
  </target>
 </lock>
</rpc>]]>]]>
```

3. Discard uncommitted changes:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="232">
 <discard-changes/>
</rpc>]]>]]>
```

4. Configure the DHCP server:

```
<rpc message-id="233"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <edit-config>
  <target>
   <candidate/>
  </target>
  <config>
   <services xmlns="http://ruggedcom.com/ns/rmf_services">
    <dhcpserver>
     <subnet>
      <name>192.168.3.0/24</name>
      <network-ip>192.168.3.0/24</network-ip>
      <options>
       <iprange>
        <start>192.168.3.30</start>
        <end>192.168.3.35</end>
       </iprange>
      </options>
     </subnet>
    </dhcpserver>
   </services>
  </config>
 </edit-config>
</rpc>]]>]]>
```

5. Commit the changes:

```
<rpc message-id="234"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <commit/>
</rpc>]]>]]>
```

6. Unlock the candidate configuration:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="235">
 <unlock>
  <target>
   <candidate/>
  </target>
 </unlock>
</rpc>]]>]]>
```

7. Unlock the running configuration:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="236">
 <unlock>
  <target>
   <running/>
```

```
  </target>
 </unlock>
</rpc>]]>]]>
```

Section 8.37

Configure the DHCP Server Port Listening for DHCP Client Requests

In this example, multiple <rpc> requests configure the DHCP listen port in the candidate configuration and then commit the changes.

This example shows the recommended procedure for making configuration changes on a device.

Procedure: Configure the DHCP Server Port Listening for DHCP Client Requests

1. Lock the running configuration:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="230">
 <lock>
  <target>
   <running/>
  </target>
 </lock>
</rpc>]]>]]>
```

2. Lock the candidate configuration:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="231">
 <lock>
  <target>
   <candidate/>
  </target>
 </lock>
</rpc>]]>]]>
```

3. Discard uncommitted changes:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="232">
 <discard-changes/>
</rpc>]]>]]>
```

4. Configure the DHCP Server listen interface:

```
<rpc message-id="233"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <edit-config>
  <target>
   <candidate/>
  </target>
  <config>
   <services xmlns="http://ruggedcom.com/ns/rmf_services">
    <dhcpserver>
     <interface>
      <name>switch.0021</name>
     </interface>
    </dhcpserver>
   </services>
  </config>
 </edit-config>
</rpc>]]>]]>
```

5. Commit the changes:

```
<rpc message-id="234"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <commit/>
</rpc>]]>]]>
```

6. Unlock the candidate configuration:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="235">
 <unlock>
  <target>
   <candidate/>
  </target>
 </unlock>
</rpc>]]>]]>
```

7. Unlock the running configuration:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="236">
 <unlock>
  <target>
   <running/>
  </target>
 </unlock>
</rpc>]]>]]>
```

Section 8.38

Enabling the DHCP Server Service

In this example, multiple `<rpc>` requests enable the DHCP server service in the candidate configuration and then commit the changes.

This example shows the recommended procedure for making configuration changes on a device.

Procedure: Enabling the DHCP Server Service

1. Lock the running configuration:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="230">
 <lock>
  <target>
   <running/>
  </target>
 </lock>
</rpc>]]>]]>
```

2. Lock the candidate configuration:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="231">
 <lock>
  <target>
   <candidate/>
  </target>
 </lock>
</rpc>]]>]]>
```

3. Discard uncommitted changes:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="232">
 <discard-changes/>
```

```
</rpc>]]>]]>
```

4. Enable the DHCP Server service:

```
<rpc message-id="233"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <edit-config>
  <target>
   <candidate/>
  </target>
  <config>
   <services xmlns="http://ruggedcom.com/ns/rmf_services">
    <dhcpserver>
     <enabled/>
    </dhcpserver>
   </services>
  </config>
 </edit-config>
</rpc>]]>]]>
```

5. Commit the changes:

```
<rpc message-id="234"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <commit/>
</rpc>]]>]]>
```

6. Unlock the candidate configuration:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="235">
 <unlock>
  <target>
   <candidate/>
  </target>
 </unlock>
</rpc>]]>]]>
```

7. Unlock the running configuration:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="236">
 <unlock>
  <target>
   <running/>
  </target>
 </unlock>
</rpc>]]>]]>
```

Section 8.39

Disabling an Ethernet Port

In this example, multiple `<rpc>` requests disable an Ethernet port in the candidate configuration and then commit the changes.

This example shows the recommended procedure for making configuration changes on a device. This example also shows how to delete data with the**nc:operation="delete"** attribute.

Procedure: Disabling an Ethernet Port

1. Lock the running configuration:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="230">
 <lock>
```

```
  <target>
   <running/>
  </target>
 </lock>
</rpc>]]>]]>
```

2. Lock the candidate configuration:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="231">
 <lock>
  <target>
   <candidate/>
  </target>
 </lock>
</rpc>]]>]]>
```

3. Discard uncommitted changes:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="232">
 <discard-changes/>
</rpc>]]>]]>
```

4. Disable the port with the**nc:operation="delete"** attribute:

```
<rpc message-id="233"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <edit-config>
  <target>
   <candidate/>
  </target>
  <config>
   <interface xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0" xmlns="http://ruggedcom.com/ns/
rmf_if">
    <eth>
     <slot>lm4</slot>
     <port>1</port>
     <enabled nc:operation="delete"/>
    </eth>
   </interface>
  </config>
 </edit-config>
</rpc>]]>]]>
```

5. Commit the changes:

```
<rpc message-id="234"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <commit/>
</rpc>]]>]]>
```

6. Unlock the candidate configuration:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="235">
 <unlock>
  <target>
   <candidate/>
  </target>
 </unlock>
</rpc>]]>]]>
```

7. Unlock the running configuration:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="236">
 <unlock>
  <target>
   <running/>
```

```
  </target>
 </unlock>
</rpc>]]>]]>
```

Section 8.40

Enabling an Ethernet Port

In this example, multiple <rpc> requests enable an Ethernet port in the candidate configuration and then commit the changes.

This example shows the recommended procedure for making configuration changes on a device.

Procedure: Enabling an Ethernet Port

1. Lock the running configuration:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="230">
 <lock>
  <target>
   <running/>
  </target>
 </lock>
</rpc>]]>]]>
```

2. Lock the candidate configuration:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="231">
 <lock>
  <target>
   <candidate/>
  </target>
 </lock>
</rpc>]]>]]>
```

3. Discard uncommitted changes:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="232">
 <discard-changes/>
</rpc>]]>]]>
```

4. Enable the Ethernet port:

```
<rpc message-id="233"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <edit-config>
  <target>
   <candidate/>
  </target>
  <config>
   <interface xmlns="http://ruggedcom.com/ns/rmf_if">
    <eth>
     <slot>lm4</slot>
     <port>1</port>
     <enabled/>
    </eth>
   </interface>
  </config>
 </edit-config>
</rpc>]]>]]>
```

5. Commit the changes:

```
<rpc message-id="234"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <commit/>
</rpc>]]>]]>
```

6. Unlock the candidate configuration:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="235">
 <unlock>
  <target>
   <candidate/>
  </target>
 </unlock>
</rpc>]]>]]>
```

7. Unlock the running configuration:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="236">
 <unlock>
  <target>
   <running/>
  </target>
 </unlock>
</rpc>]]>]]>
```

Section 8.41

Checking an IP Address on a Specific Port using the Interfaces Namespace

In this example, a single `<rpc>` request retrieves the IP address from a specified port.

This example shows the typical procedure for querying data from the running configuration.

Procedure: Checking an IP Address on a Specific Port using the Interfaces Namespace

- Request the data from the running configuration:

```
<rpc message-id="2"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <get>
  <filter type="subtree">
   <interfaces xmlns="http://ruggedcom.com/ns/rmf_ifs">
    <ip>
     <name>fe-4-1</name>
     <ipv4>
      <address>
       <local></local>
      </address>
     </ipv4>
    </ip>
   </interfaces>
  </filter>
 </get>
</rpc>]]>]]>
```

Section 8.42

Configuring Frame Relay on a WAN Port

In this example, multiple <rpc> requests configure frame relay on a WAN port in the candidate configuration and then commit the changes.

This example shows the recommended procedure for making configuration changes on a device.

Procedure: Configuring Frame Relay on a WAN Port

1. Lock the running configuration:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="230">
 <lock>
  <target>
   <running/>
  </target>
 </lock>
</rpc>]]>]]>
```

2. Lock the candidate configuration:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="231">
 <lock>
  <target>
   <candidate/>
  </target>
 </lock>
</rpc>]]>]]>
```

3. Discard uncommitted changes:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="232">
 <discard-changes/>
</rpc>]]>]]>
```

4. Configure the Frame Relay parameters:

```
<rpc message-id="233"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
<edit-config>
  <target>
   <candidate/>
  </target>
  <config>
   <interface xmlns="http://ruggedcom.com/ns/rmf_if">
    <wan>
     <slot>lm3</slot>
     <port>1</port>
     <t1>
      <t1params>
       <clock>normal</clock>
      </t1params>
      <channel>
       <channelnumber>01</channelnumber>
       <ts>all</ts>
       <connection>
        <framerelay>
         <param>
          <station>cpe</station>
         </param>
         <dlci>
          <id>16</id>
         </dlci>
        </framerelay>
```

```
       </connection>
      </channel>
     </t1>
    </wan>
   </interface>
  </config>
 </edit-config>
</rpc>]]>]]>
```

5. Commit the changes:

```
<rpc message-id="234"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <commit/>
</rpc>]]>]]>
```

6. Unlock the candidate configuration:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="235">
 <unlock>
  <target>
   <candidate/>
  </target>
 </unlock>
</rpc>]]>]]>
```

7. Unlock the running configuration:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="236">
 <unlock>
  <target>
   <running/>
  </target>
 </unlock>
</rpc>]]>]]>
```

Section 8.43

Retreiving All Data From Running Database Including Default Values

In this example, a single `<rpc>` request retrieves information (including default values) from a specified configuration on a running database.

This example shows the recommended procedure for querying data from a running database.

Procedure: Retrieving Data From a Running Database

- Request data from the running databse:

```
<rpc message-id="1"  xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" with-defaults="true">
 <get-config xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <source>
   <running/>
  </source>
 </get-config>
</rpc>
```

Section 8.44

Retreiving All Data From Running Database Including Default Tags and Values

In this example, a single <rpc> request retrieves information (including default tags and values) from a specified configuration on a running database.

This example shows the recommended procedure for querying data from a running database.

Procedure: Retrieving Data From a Running Database

- Request data from the running databse:

```
<rpc message-id="1"  xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <get-config xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <source>
   <running/>
  </source>
  <with-defaults xmlns="urn:ietf:params:xml:ns:yang:ietf-netconf-with-defaults">report-all-tagged</
with-defaults>
 </get-config>
</rpc>
```

Section 8.45

Changing a User's Password

In this example, a single <rpc> request changes the password for a specific user.

This example shows the typical procedure for changing user profiles from the running configuration.

Procedure: Changing a User's Password

1. Discard uncommitted changes:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="232">
 <discard-changes/>
</rpc>]]>]]>
```

IMPORTANT!
The password must be provided in hash format. Use a utility such as mkpasswd (available on most Linux distributions) to generate a hashed password. For a Windows-based utility, contact Siemens Customer Service.

2. Lock the running configuration:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="230">
 <lock>
  <target>
   <running/>
  </target>
 </lock>
</rpc>]]>]]>
```

3. Lock the candidate configuration:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="231">
 <lock>
```

```
  <target>
   <candidate/>
  </target>
 </lock>
</rpc>]]>]]>
```

4. Configure a user's password:

```
<rpc message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <edit-config>
  <target>
   <candidate/>
  </target>
  <config>
   <admin xmlns="http://ruggedcom.com/ns/rmf_admin">
    <users>
     <user-id>
      <name>{userName}</name>
      <password>{hashPassword}</password>
      <role>{userRole}</role>
     </user-id>
    </users>
   </admin>
  </config>
 <edit-config>
</rpc>]]>]]>
```

5. Commit the changes:

```
<rpc message-id="234"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <commit/>
</rpc>]]>]]>
```

6. Unlock the candidate configuration:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="235">
 <unlock>
  <target>
   <candidate/>
  </target>
 </unlock>
</rpc>]]>]]>
```

7. Unlock the running configuration:

```
<rpc xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="236">
 <unlock>
  <target>
   <running/>
  </target>
 </unlock>
</rpc>]]>]]>
```

Section 8.46

Displaying the Status

Displays the status of the running IPSec service. This action does not take any parameters.

```
<rpc message-id="2"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
```

```
 <get>
 <filter type="subtree">
   <tunnel xmlns="http://ruggedcom.com/ns/rmf_iftunnel">
    <ipsec>
     <status>
     </status>
    </ipsec>
   </tunnel>
  </filter>
 </get>
</rpc>]]>]]>
```

Section 8.47

Installing a Certificate to an IPSec Connection

This action uploads an IPSec certificate to the device. The certificate must be located at a network location accessible to the device.

```
<rpc message-id="233"
 xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
<edit-config>
 <target><candidate/></target>
  <config><tunnel xmlns="http://ruggedcom.com/ns/rmf_iftunnel">
   <ipsec>
    <connection>
     <name>{name}</name>
     <startup>{startup}</startup>
     <authenticate>{authenticate}</authenticate>
     <connection-type>{connection type}</connection-type>
     <monitor-interface>{monitor interface}</monitor-interface>
     <left>
      <public-ip>
       <type>{ip type}</type>
       <value>{ip value}</value>
      </public-ip>
      <subnet>
       <network>{network}</network>
      </subnet>
      <key>
       <type>{key type}</type>
       <certificate>{certificate}</certificate>
      </key>
      <nexthop>
       <type>{nexthop type}</type>
       <value>{nexthop value}</value>
      </nexthop>
     </left>
     <right>
      <public-ip>
       <type>{ip type}</type>
       <value>{ip value}</value>
      </public-ip>
      <subnet>
       <network>{network}</network>
      </subnet>
      <key>
       <type>{key type}</type>
       <certificate>{certificate}</certificate>
      </key>
      <nexthop>
       <type>{nexthop type}</type>
```

```
        <value>{nexthop value}</value>
       </nexthop>
      </right>
     </connection>
    </ipsec>
  </tunnel></config>
 </edit-config>
</rpc>]]>]]>
```

- `{name}`

  The name of the IPSec tunnel.

- `{startup}`

  The action to take when IPSec is initialized. Must be one of the following: **`add`**, **`default`**, **`ignore`**, **`route`**, or **`start`**.

- `{authenticate}`

  The authentication method of the IPSec tunnel. Must be one of the following: **`rsasig`**, **`secret`**, or **`default`**.

- `{connection type}`

  The IPSec connection type. Must be one of the following: **`tunnel`**, **`transport`**, **`passthrough`**, or **`default`**.

- `{monitor interface}`

  The interface to monitor where the IPSec tunnel is running/established. For example: **`fe-cm-1`**, **`switch.0001`**, etc.

- `{ip type}`

  The public ip address type of the IPSec tunnel. Must be one of the following: **`address`**, **`any`**, **`default-route`** **`hostname`**, or **`none`**.

- `{ip value}`

  The value is based on the selected {ip type} value. For example, if **`address`** is chosen as the ip type, an ip address is defined here.

- `{network}`

  The local network address or subnet value of the IPSec tunnel (e.g 192.168.0.0/24).

- `{key type}`

  The IPSec tunnel will be established using one of the following: **`certificates`**, **`certificates-any`**, **`none,`** or **`rsasig`**.

- `{certificate}`

  The name of the certificate already installed on the system.

- `{nexthop type}`

  The next hop to the other system. Must be one of the following: **`address`**, **`default`** or **`default-route`**.

- `{nexthop value}`

  The IP address of the next hop that can be used to reach the destination network. The value is defined based on the selected {nexthop type} value. For example, if **`address`** is chosen as the nexthop type, an ip address is defined here.

Section 8.48
Installing a CA Certificate

This action uploads a Certificate Authority (CA) certificate to the device. The CA certificate must be located at a network location accessible to the device. Parameters include `<name>`, `<ca-name>`, and `<private-key-name>`.

```
<rpc message-id="233"
 xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <edit-config>
  <target><candidate/></target>
   <config>
    <security xmlns="http://ruggedcom.com/ns/rmf_security">
     <crypto>
      <certificate>
       <name>{name}</name>
       <ca-name>{ca name}</ca-name>
       <private-key-name>{private key name}</private-key-name>
       <contents>
        -----BEGIN CERTIFICATE-----
        MIIC8jCCAlugAwIBAgIBATANBgkqhkiG9w0BAQUFADCBiTELMAkGA1UEBhMCQ0Ex...
        IsAFEeX2iShlXT7OSYqS771RFFSp1dzirAcndiFeUUzXm5Gj8P4=
        -----END CERTIFICATE-----
       </contents>
      </certificate>
     </crypto>
    </security>
   </config>
 </edit-config>
</rpc>]]>]]>
```

- `{name}`

  The name of the CA certificate.

- `{ca name}`

  The name of the Certificate Authority.

- `{private key name}`

  The name of the private key that corresponds to the CA certificate.

Section 8.49
Configuring a Signed CA Certificate

This action configures a signed CA certificate . The CA certificate must be located at a network location accessible to the device. Parameters include `<name>`, `<ca-name>`, and `<private-key-name>`.

```
<rpc message-id="233" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <edit-config>
  <target>
   <candidate/>
    </target>
     <config>
      <security xmlns="http://ruggedcom.com/ns/rmf_security">
       <crypto>
        <certificate>
         <name>{name}</name>
         <ca-name>{ca name}</ca-name>
```

```
        <private-key-name>{private key name}</private-key-name>
        <contents>
         -----BEGIN CERTIFICATE-----
         MIIC8jCCAlugAwIBAgIBATANBgkqhkiG9w0BAQUFADCBiTELMAkGA1UEBhMCQ0Ex...
         IsAFEeX2iShlXT7OSYqS771RFFSp1dzirAcndiFeUUzXm5Gj8P4=
         -----END CERTIFICATE-----
        </contents>
       </certificate>
      </crypto>
     </security>
    </config>
   </edit-config>
  </rpc>]]>]]>
```

- `{name}`

  The name of the `.pem` file.

- `{ca name}`

  The name of the Certificate Authority.

- `{private key name}`

  The name of the private key that corresponds to the CA certificate.

Section 8.50

Installing a Private Key to a Signed CA Certificate

This action installs a private key to a signed CA certificate. The certificate must be located at a network location accessible to the device. Parameters include `<name>` and `<algorithm>`.

```
<rpc message-id="233" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
<edit-config>
  <target>
   <candidate/>
    </target>
     <config>
      <security xmlns="http://ruggedcom.com/ns/rmf_security">
       <crypto>
        <private-key>
         <name>{name}</name>
         <algorithm>{algorithm}</algorithm>
         <contents>
          -----BEGIN RSA PRIVATE KEY-----
          MIICXQIBAAKBgQDCI7Xy6OF1XVcQNTSqTyLDKJ+knhYQawrgRoE4Q677q9taftee...
          dsCQmC2smAtdrfY/VaGJrX6ZdiUyRcxNLsDNBoNQcZQH
          -----END RSA PRIVATE KEY-----
         </contents>
        </private-key>
       </crypto>
      </security>
     </config>
    </edit-config>
  </rpc>]]>]]>
```

- `{name}`

  The name of the private key.

- `{algorithm}`

  The type of private key. Must be one of the following: dsa, rsa, or ssh-rsa.

Section 8.51
Installing a CRL File

This action installs a Certificate Revocation List (CRL) file to the device. The `.crl` file must be located at a network location accessible to the device.

```
<rpc message-id="233"
 xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <edit-config>
  <target>
   <candidate/>
    </target>
     <config>
      <security xmlns="http://ruggedcom.com/ns/rmf_security">
       <crypto>
        <ca><name>{name}</name>
         <key-cert-sign-certificate>
          -----BEGIN CERTIFICATE-----
          MIID5zCCAs+gAwIBAgIJAK85lF/lcPaKMA0GCSqGSIb3DQEBCwUAMIGJMQswCQYD...
          SrCK8Rwp9S89hiwD5FNlQcIvYnjacNg8G9l8CNiLF71BK4lEroPk3ZVTpw==
          -----END CERTIFICATE-----
         </key-cert-sign-certificate>
         <crl>
          -----BEGIN X509 CRL-----
          MIIB4zCBzAIBATANBgkqhkiG9w0BAQUFADCBiTELMAkGA1UEBhMCQ0ExCzAJBgNV...
          7KJR/xXHQPNhIKiTqwyzJm32rih1TNWh7sB26Hh2armPOlq3vqEsR9vdo/g5hF18
          0j4fMlYlWA==
          -----END X509 CRL-----
        </crl>
       </ca>
      </crypto>
     </security>
    </config>
  </edit-config>
 </rpc>]]>]]>
```

- `{name}`

  The name of the `.crl` file.

Section 8.52
Removing a Certificate

This action removes the specified certificate from the device. Specify the certificate name in the `<name>` element. This action does not take any parameters.

```
<rpc message-id="233"
 xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <edit-config>
  <target>
   <candidate/>
    </target>
     <config>
      <security xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0"
      xmlns="http://ruggedcom.com/ns/rmf_security">
       <crypto>
        <certificate nc:operation="delete"><name>{name}</name><contents>
         -----BEGIN CERTIFICATE-----
         MIIDCDCCAnGgAwIBAgIJAPl3LLRHpm/cMA0GCSqGSIb3DQEBBQUAMIGcMQswCQYD...
```

```
        ZptpoW/N2920tXQvsjD4SG+EoCPilKD63vbb54UFh/10SRIlIUplCDu1uXNvI3Pe
        u+Kh+vRZz8IqXtI0
        -----END CERTIFICATE-----
       </contents></certificate>
      </crypto>
     </security>
    </config>
   </edit-config>
  </rpc>]]>]]>
```

- `{name}`

  The name of the certificate to remove.

Section 8.53
Removing a CA certificate

This action removes the specified CA certificate from the device. Specify the certificate name in the `<name>` element. This action does not take any parameters.

```
<rpc message-id="233"
 xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <edit-config>
   <target>
    <candidate/>
     </target>
      <config>
       <security xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0"
       xmlns="http://ruggedcom.com/ns/rmf_security">
        <crypto>
         <ca nc:operation="delete"><name>{name}</name>
          <key-cert-sign-certificate>
           -----BEGIN CERTIFICATE-----
           MIID5zCCAs+gAwIBAgIJAK85lF/lcPaKMA0GCSqGSIb3DQEBCwUAMIGJMQswCQYD...
           SrCK8Rwp9S89hiwD5FNlQcIvYnjacNg8G9l8CNiLF71BK4lEroPk3ZVTpw==
           -----END CERTIFICATE-----
          </key-cert-sign-certificate>
         </ca>
        </crypto>
       </security></config>
      </edit-config>
     </rpc>]]>]]>
```

- `{name}`

  The name of the CA certificate to remove.

Section 8.54
Removing a CRL File

This action removes the specified `.crl` file from the device. Specify the `.crl` file name in the `<name>` element. This action does not take any parameters.

```
<rpc message-id="233"
 xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
 <edit-config>
  <target>
```

```
  <candidate/>
   </target>
    <config><security xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0"
    xmlns="http://ruggedcom.com/ns/rmf_security">
     <crypto>
      <ca><name>{crlFileName}</name>
       <crl nc:operation="delete">
        -----BEGIN X509 CRL-----
        MIIB4zCBzAIBATANBgkqhkiG9w0BAQUFADCBiTELMAkGA1UEBhMCQ0ExCzAJBgNV...
        7KJR/xXHQPNhIKiTqwyzJm32rih1TNWh7sB26Hh2armPOlq3vqEsR9vdo/g5hF18
        0j4fMlYlWA==
        -----END X509 CRL-----
       </crl>
      </ca>
     </crypto>
    </security>
   </config>
  </edit-config>
</rpc>]]>]]>
```

- `{crlFileName}`

  The name of the `.crl` file to remove.

9 NETCONF XML Elements

Section 9.1

]]>]]>

Description: Indicates the end of an XML document. The]]>]]> sequence must appear at the end of each XML document sent by the client and server.

Example: Using]]>]]> following an <rpc> element:

```
<rpc>
  <close-session/>
</rpc>
]]>]]>
```

Section 9.2

<close-session/>

Description: Requests the graceful termination of a NETCONF session.

When a NETCONF server receives a <close-session> request, it does the following:

- gracefully closes the session
- releases any locks and resources associated with the session
- gracefully closes any associated connections
- ignores any NETCONF requests received after the <close-session> request

Response: If the NETCONF device can complete the request, it sends an <rpc-reply> document containing the <ok> element.

If the NETCONF device cannot complete the request, it sends an <rpc-reply> document containing the <rpc-error> element.

Example: Using <close-session/> to close a NETCONF session:

```
<rpc>
  <close-session/>
</rpc>
]]>]]>
```

Section 9.3

<commit>

Description: Commits the changes made to the candidate configuration, applying the changes to the device's currently running configuration.

Commits can be immediate, or can require confirmation:

- To commit the changes immediately, issue an empty <commit/> tag: the changes immediately apply to the currently running configuration.
- To require confirmation of the changes, issue the <confirmed/> tag within the <commit/> tag: the changes appear in the currently running configuration, but are rolled back if they are not confirmed within a timeout period. The default timeout period is 10 minutes. To specify a different timeout period, use the <confirm-timeout> tag within the <commit/> tag. To confirm the commit before the timeout period expires, issue an empty <commit/> tag.

Parameters: <confirmed/> : requires the commit to be confirmed within a configurable timeout period. If a timeout period is not specified with the optional <confirm-timeout> tag, the default period is 10 minutes. To confirm the commit within the timeout period, issue an empty <commit/> tag.

<confirm-timeout> : specifies the timeout period, in minutes, during which you can confirm a commit. If the commit is not confirmed within the timeout period, the configuration rolls back to the previously active configuration.

Example: To commit changes immediately:

```
<rpc>
  <commit/>
</rpc>
]]>]]>
```

To commit changes and require a commit confirmation with the default timeout period:

```
<rpc>
  <commit/><confirmed/></commit>
</rpc>
]]>]]>
```

To commit changes and require a commit confirmation within a 30 minute timeout period:

```
<rpc>
  <commit><confirmed/><confirm-timeout>30</confirm-timeout></commit>
</rpc>
]]>]]>
```

Section 9.4

<copy-config>

Description: Creates or replaces a specified <target> configuration with a specified <source> configuration. If the <target> configuration exists, it is overwritten. If the <target> configuration does not exist, a new configuration is created.

Parameters: <target> : specifies the target configuration to create or to overwrite. Valid values: running| candidate.

<source> : a container for the configuration source file to copy.

<url> : specifies the URL of the configuration source file to copy.

Response: If the NETCONF device can complete the request, it sends an <rpc-reply> document containing the <ok> element.

If the NETCONF device cannot complete the request, it sends an <rpc-reply> document containing the <rpc-error> element.

Example: To copy a configuration and make it the candidate configuration:

```
<rpc>
  <copy-config>
    <target>
      <candidate/>
    </target>
    <source>
      <url>
        <https://user@example.com:passphrase/path/filename.txt>
      </url>
    </source>
  <copy-config>
</rpc>
]]>]]>
```

Section 9.5

<data>

Description: Encloses configuration data returned from the device.

Example: Response from a device when queried for the system name:

```
<rpc-reply message-id="2" with-defaults="true"
 xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <data>
    <admin xmlns="http://ruggedcom.com/ns/rmf_admin">
      <system-name>System Name</system-name>
    </admin>
  </data>
</rpc-reply>
```

Section 9.6

<discard-changes>

Description: Discards changes to the candidate configuration and reverts the candidate configuration to the currently running configuration.

Example: To discard changes to the candidate configuration:

```
<rpc>
  <discard-changes/>
</rpc>
]]>]]>
```

Section 9.7

<edit-config>

Description: Changes the specified data in a specified configuration.

Parameters: <target> : the configuration to edit: candidate or running.

<config> : identifies the configuration segments to edit. The filter element contains elements describing the configuration parameters to set and the date values to set in the configuration.

Response: If the NETCONF device can complete the request, it sends an <rpc-reply> document containing the <data> element and results of the query.

If the NETCONF device cannot complete the request, it sends an <rpc-reply> document containing the <rpc-error> element.

Example: To change the system name parameter in the running configuration:

```
<rpc message-id="233" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <edit-config>
    <target>
      <running/>
    </target>
    <config>
      <admin xmlns="http://ruggedcom.com/ns/rmf_admin"><system-name>Lorem Ipsum</
system-name></admin>
    </config>
  </edit-config>
</rpc>]]>]]>
```

Section 9.8

<error-info>

Description: In an <rpc-error> element, <error-info> contains information describing an error returned by the NETCONF server. Elements within <error-info> may indicate specific errors. For more information on NETCONF errors, see Internet Engineering Task Force RFC 6241 [http://tools.ietf.org/html/rfc6241].

Example: An <rpc-reply> response with <error-info> indicating an error due to attribute and element problems:

```
<rpc-reply>
  <rpc-error>
    <error-type>rpc</error-type>
    <error-tag>missing-attribute</error-tag>
    <error-severity>error</error-severity>
    <error-info>
      <bad-attribute>message-id</bad-attribute>
      <bad-element>rpc</bad-element>
    </error-info>
  </rpc-error>
</rpc-reply>
]]>]]>
```

Section 9.9

<get-config>

Description: Requests all or part of a specified configuration.

Parameters: <source> : the configuration to query: candidate or running.

<filter> : identifies the configuration segments to retrieve. The filter element contains elements describing the configuration segment to return. For more information, see Filtering.

Response: If the NETCONF device can complete the request, it sends an <rpc-reply> document containing the <data> element and results of the query.

If the NETCONF device cannot complete the request, it sends an <rpc-reply> document containing the <rpc-error> element.

Example: To return configuration data from the chassis namespace:

```
<rpc message-id="2" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <get-config>
    <source>
      <running/>
    </source>
    <filter type="subtree">
      <chassis xmlns="http://ruggedcom.com/ns/rmf_chassis"></chassis>
    </filter>
  </get-config>
</rpc>]]>]]>
```

Section 9.10

<hello>

Description: Lists the capabilities of the NETCONF server and client. When connecting to the device, the device sends a <hello> message containing its NETCONF capabilities and a session-id. The client connecting to the device must also send a <hello> message, listing at least the base NETCONF capability. The client's <hello> message must not contain a session-id.

Parameters: <capabilities> : contains one or more <capability> elements.

<capability> : contains the uniform resource identifier (URI) for a single NETCONF capability. Standard NETCONF capabilities appear with a universal resource name (URN). Vendor-defined NETCONF capabilities appear with either a URN or universal resource locator (URL).

<session-id> : a session identifier returned by the NETCONF server. A NETCONF client must not return a <session-id> element in its hello message. If the client returns a <session-id> element, the server terminates the session.

Example: A <hello> message returned from a device:

```
<?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</capability>
<capability>urn:ietf:params:netconf:capability:candidate:1.0</capability>
<capability>urn:ietf:params:netconf:capability:confirmed-commit:1.0</capability>
<capability>urn:ietf:params:netconf:capability:confirmed-commit:1.1</capability>
<capability>urn:ietf:params:netconf:capability:xpath:1.0</capability>
<capability>urn:ietf:params:netconf:capability:url:1.0?scheme=ftp,sftp,file</
capability>
<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:rollback-on-error:1.0</capability>
<capability>urn:ietf:params:netconf:capability:notification:1.0</capability>
<capability>urn:ietf:params:netconf:capability:interleave:1.0</capability>
<capability>urn:ietf:params:netconf:capability:partial-lock:1.0</capability>
<capability>http://tail-f.com/ns/netconf/with-defaults/1.0</capability>
<capability>http://tail-f.com/ns/netconf/actions/1.0</capability>
<capability>http://tail-f.com/ns/netconf/commit/1.0</capability>
<capability>urn:ietf:params:netconf:capability:with-defaults:1.0?basic-
mode=trim&also-supported=report-all-tagged</capability>
<capability>urn:ietf:params:xml:ns:yang:ietf-netconf-with-defaults?
revision=2010-11-11&module=ietf-with-defaults</capability>
<capability>http://ruggedcom.com/ns/rmf?module=rmf&revision=2012-03-07</capability>
<capability>http://ruggedcom.com/ns/rmf_admin?
module=rmf_admin&revision=2012-03-07</capability>
<capability>http://ruggedcom.com/ns/rmf_chassis?
module=rmf_chassis&revision=2012-03-07</capability>
<capability>http://ruggedcom.com/ns/rmf_events?
module=rmf_events&revision=2012-03-07</capability>
<capability>http://ruggedcom.com/ns/rmf_global?
module=rmf_global&revision=2012-03-07</capability>
<capability>http://ruggedcom.com/ns/rmf_if?module=rmf_if&revision=2012-03-07</
capability>
<capability>http://ruggedcom.com/ns/rmf_ifs?module=rmf_ifs&revision=2012-03-07</
capability>
<capability>http://ruggedcom.com/ns/rmf_iftunnel?
module=rmf_iftunnel&revision=2012-03-07</capability>
<capability>http://ruggedcom.com/ns/rmf_ip?module=rmf_ip&revision=2012-03-07</
capability>
<capability>http://ruggedcom.com/ns/rmf_qos?module=rmf_qos&revision=2012-03-07</
capability>
<capability>http://ruggedcom.com/ns/rmf_routing?
module=rmf_routing&revision=2012-03-07</capability>
<capability>http://ruggedcom.com/ns/rmf_security?
module=rmf_security&revision=2012-03-07</capability>
<capability>http://ruggedcom.com/ns/rmf_services?
module=rmf_services&revision=2012-03-07</capability>
<capability>http://tail-f.com/yang/common-monitoring?module=tailf-common-
monitoring&revision=2011-09-22</capability>
<capability>http://tail-f.com/yang/confd-monitoring?module=tailf-confd-
monitoring&revision=2011-09-22</capability>
<capability>http://tail-f.com/yang/netconf-monitoring?module=tailf-netconf-
monitoring&revision=2011-09-22</capability>
<capability>urn:ietf:params:xml:ns:yang:ietf-inet-types?module=ietf-inet-
types&revision=2010-09-24</capability>
<capability>urn:ietf:params:xml:ns:yang:ietf-netconf-monitoring?module=ietf-
netconf-monitoring&revision=2010-10-04</capability>
<capability>urn:ietf:params:xml:ns:yang:ietf-yang-types?module=ietf-yang-
types&revision=2010-09-24</capability>
</capabilities>
<session-id>159</session-id></hello>]]>]]>
```

The minimum <hello> message required from a NETCONF client:

```
<hello xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <capabilities>
    <capability>urn:ietf:params:netconf:base:1.0</capability>
  </capabilities>
</hello>]]>]]>
```

Section 9.11
<kill-session>

Description: Terminates a specified NETCONF session, cancelling any operations in progress and releasing all locks, resources, and connections for the session.

<kill-session> does not roll back the configuration or state changes made by the configuration being terminated. If the session being terminated is performing a confirmed commit when the <kill-session> is issued, the NETCONF server restores the configuration to its state before the confirmed commit was issued.

To kill a session, you need to know its <session-id>. To find a session's <session-id>, attempt to <lock> or <unlock> the session. The <session-id> is reported in the <rpc-error> message received from the unsuccessful <lock> or <unlock> attempt.

Parameters: <session-id> : the unique identifier for a session.

Response: If the NETCONF device can complete the request, it sends an <rpc-reply> document containing the <ok> element.

If the NETCONF device cannot complete the request, it sends an <rpc-reply> document containing the <rpc-error> element.

Example: To kill a session:

```
<rpc message-id="101" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <kill-session>
    <session-id>4</session-id>
  </kill-session>
</rpc>
```

Section 9.12
<lock>

Description: Locks the specified configuration, preventing other NETCONF sessions and other services, such as the web interface and command line interface, from editing the session. Other sessions may read a locked session, but cannot edit it.

The <lock> operation fails if the configuration is already locked by the current or another session, or if the configuration has been modified by the current session but not yet committed.

Only the session performing the <lock> operation can unlock the configuration with the <unlock> operation. If the session is terminated before the <unlock> operation is performed, the configuration is automatically unlocked.

Parameters: <target> : the configuration to lock: <candidate/> or <running/>

Response: If the NETCONF device can complete the request, it sends an <rpc-reply> document containing the <ok> element.

If the NETCONF device cannot complete the request, it sends an <rpc-reply> document containing the <rpc-error> element.

Example: To lock the running configuration:

```
<rpc message-id="104" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
```

```
  <lock>
    <target>
      <running/>
    </target>
  </lock>
</rpc>
]]>]]>
```

Section 9.13
<ok/>

Description: Appears in an <rpc-reply> message to indicate successful completion of a NETCONF request.

Example: An <rpc-reply> message indicating the successful completion of a NETCONF request:

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

Section 9.14
<rpc>

Description: Encloses NETCONF requests sent to the NETCONF server. The `netconf:base` namespace declaration and the `message-id` attributes are mandatory.

The `message-id` attribute is an arbitrary string identifying the request. The `message-id` string is returned as part of the <rpc-reply> message in response to the request, helping to map the response to the request. The `message-id` strings do not need to be unique within a session.

Example: The <rpc> element in a request, and the resulting <rpc-reply> message. Note the <message-id> attribute in the request and the reply.

```
<rpc message-id="103" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <validate>
    <source>
      <running/>
    </source>
  </validate>
</rpc>
]]>]]>
<?xml version="1.0" encoding="UTF-8"?>

<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="103"><ok/></
rpc-reply>]]>]]>
```

Section 9.15

<rpc-error>

Description: Indicates that the NETCONF server encountered an error processing an <rpc> request. The <rpc-error> element appears within <rpc-request> messages.

For more information on NETCONF errors, see Internet Engineering Task Force RFC 6241 [http://tools.ietf.org/html/rfc6241].

Parameters: The <rpc-error> element includes the following information:

<error-type> : indicates the conceptual layer where the error occurred: transport | rpc | protocol | application

<error-tag> : identifies the error condition.

<error-severity> : indicates the severity of the error: error | warning

<error-app-tag> : indicates the data-model or implementation error condition, if there is one. This element does not appear if no application error tag is associated with the error condition.

<error-path> : shows the XPath to the element associated with the error, if there is one. This element does not appear if no element is associated with the error condition.

<error-message> : shows a human-readable error message describing the error condition. This element does not appear if no error message is available for the error condition.

<error-info> : shows protocol or data-model error content. This element does not appear if no error content is available for the error condition.

Example: An <rpc-error> in response to an <rpc> request:

```
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="103">
  <rpc-error>
    <error-type>application</error-type>
    <error-tag>operation-failed</error-tag>
    <error-severity>error</error-severity>
    <error-path xmlns:rmf_admin="http://ruggedcom.com/ns/rmf_admin">
/rmf_admin:admin/rmf_admin:authentication
</error-path>
    <error-message xml:lang="en">/admin/authentication: admin/timezone must be
 set</error-message>
    <error-info>
      <bad-element>authentication</bad-element>
    </error-info>
  </rpc-error>
</rpc-reply>
]]>]]>
```

Section 9.16

<rpc-reply>

Description: Contains the results of an <rpc> request. The <rpc-reply> may contain returned data, the <ok/> element indicating the successful completion of an operation request, or error information.

The `user-id` attribute contains the `user-id` string sent with the <rpc> request. The `user-id` attribute helps to map the <rpc-reply> to the original <rpc> message.

Examples: An <rpc-reply> message containing data from the NETCONF server:

```
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="101">
  <data>
    <interface xmlns="http://ruggedcom.com/ns/rmf_if">
      <modem>
        <slot>lm2</slot>
        <port>1</port>
      </modem>
      <wan>
        <slot>lm5</slot>
        <port>1</port>
      </wan>
      <eth>
        <slot>lm3</slot>
        <port>1</port>
      </eth>
    </interface>
  </data>
</rpc-reply>]]>]]>
```

An <rpc-reply> message indicating the successful completion of a request:

```
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="103">
  <ok/>
</rpc-reply>]]>]]>
```

An <rpc-reply> message containing error information:

```
<rpc-reply xmlns="urn:ietf:params:xml:ns:netconf:base:1.0" message-id="104"><rpc-
error>
  <error-type>protocol</error-type>
    <error-tag>lock-denied</error-tag>
    <error-severity>error</error-severity>
    <error-info>
      <session-id>3216</session-id>
    </error-info>
  </rpc-error>
</rpc-reply>]]>]]>
```

Section 9.17

<target>

Description: Specifies the configuration on which to perform on operation. <target> is used in the <copy-config>, <delete-config>, <edit-config>, <lock>, and <unlock> operations.

Parameters: <candidate/> : specifies the candidate configuration.

<running/> : specifies the candidate configuration.

Response: If the NETCONF device can complete the request, it sends an <rpc-reply> document containing the <ok> element.

If the NETCONF device cannot complete the request, it sends an <rpc-reply> document containing the <rpc-error> element.

Example: To lock the candidate configuration:

```
<rpc>
  <lock>
    <target>
```

```
      <candidate/>
    </target>
  </lock>
</rpc>
]]>]]>
```

Section 9.18
<unlock>

Description: Releases the configuration lock placed by an earlier <lock> operation in the same NETCONF session. The specified configuration must already be locked, and only the session that performed the <lock> operation can unlock the configuration.

Parameters: <target> : the configuration to unlock: <candidate/> or <running/>

Response: If the NETCONF device can complete the request, it sends an <rpc-reply> document containing the <ok> element.

If the NETCONF device cannot complete the request, it sends an <rpc-reply> document containing the <rpc-error> element.

Example: To unlock the running configuration:

```
<rpc message-id="105" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <unlock>
    <target>
      <running/>
    </target>
  </unlock>
</rpc>
]]>]]>
```

Section 9.19
<validate>

Description: Validates the syntax of the specified configuration. After making changes to the candidate configuration, use <validate> to make sure the syntax is correct.

Parameters: <source> : specifies the configuration to validate: <candidate/> or <running/>.

Response: If the NETCONF device can complete the request, it sends an <rpc-reply> document containing the <ok> element.

If the NETCONF device cannot complete the request, it sends an <rpc-reply> document containing the <rpc-error> element. The <rpc-error> element will contain information on the syntax errors found in the configuration.

Example: To verify the candidate configuration:

```
<rpc  message-id="103" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <validate>
    <source>
      <candidate/>
    </source>
  </validate>
```

```
</rpc>
]]>]]>
```

A Uploading Files to the Device

This section describes how to upload configuration files to a ROXII device. You can upload files through the ROXII web interface and through the command line interface.

Uploading Files with the ROXII Web Interface

To upload a configuration file device through the ROXII web interface, do the following:

1. In a web browser, navigate to and log in to the device.
2. On the **Tools** tab, select **Upload**.

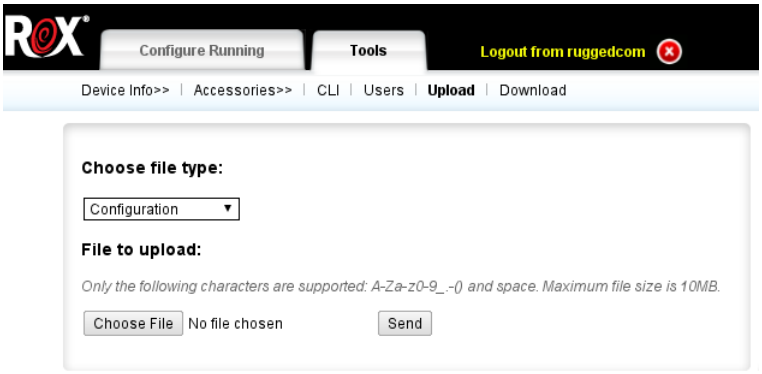

Figure 9: Upload form

3. From the **Choose file type** list, select `Configuration`.
4. Click **Choose File**. A dialog appears where you can navigate to and select a configuration file.
5. Select a configuration file and click **Open**. The name of the selected file appear beside the **Choose File** button.
6. Click **Send**. The device uploads the file.

Uploading Files with the ROXII Command Line Interface

To upload files through the command line interface, you need to know the following:

- a user name and password for the remote computer
- the IP address or hostname of the remote computer
- the directory path to the configuration file on the remote computer

- the configuration file filename

To upload a configuration file device through the ROXII command line interface, do the following:

1. In a terminal, connect to and log in to the device's command line interface.
2. Issue the file scp-config-from-url command in the following format:

```
file scp-config-from-url {user}@{host}:/{path}/{filename} {filename}
```

   - `{user}`

     A user name with access rights to the remote computer.
   - `{host}`

     The hostname or IP address of the remote computer.
   - `{path}`

     The path to the configuration file on the remote computer.
   - `{filename}`

     The filename of the configuration file.
3. When prompted, type the password to connect to the remote computer.

   For example:

```
ruggedcom# file scp-config-from-url wsmith@10.200.20.39:/c:/ruggedcom/standard_config
 standard_config
wsmith@10.200.20.39's password: {type password here; it remains hidden from view}
standard_config.txt                                           100% 7673          7.5KB/s
00:00
```

   The device uploads the file.