NETCONF	B. Lengyel
Internet-Draft	Ericsson Hungary
Intended status: Standards Track	M. Bjorklund
Expires: February 9, 2009	Tail-f Systems
	August 08, 2008

Partial Lock RPC for NETCONF
draft-ietf-netconf-partial-lock-03

Status of this Memo

Abstract

The NETCONF protocol defines the lock and unlock RPCs that lock entire configuration datastores. In some situations, a way to lock only parts of a configuration datastore is required. This document defines a capability-based extension to the NETCONF protocol for locking portions of a configuration datastore.

Table of Contents

1. Introduction

TOC

The NETCONF protocol [NETCONF] (Enns, R., “NETCONF Configuration Protocol,” December 2006.) describes the lock and unlock RPCs that operate on entire configuration datastores. Often, multiple management sessions need to be able to modify the configuration of a managed device in parallel. In these cases, locking only parts of a configuration datastore is needed. This document defines an extension to the NETCONF protocol to allow this.

The mechanism for partial locking is based on the existing XPath filtering mechanisms.

Partial locking is defined as a capability to NETCONF.

TOC

1.1. Definition of Terms

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14, [RFC2119] (Bradner, S., “Key words for use in RFCs to Indicate Requirement Levels,” March 1997.).

2. Partial Locking Capability

TOC

2.1. Overview

TOC

The :partial-lock capability indicates that the device supports the locking of its configuration with a scope smaller then a complete configuration datastore. The scope to be locked is specified by using restricted or full XPath expressions. Partial locking covers configuration data, but not state data.

The system MUST ensure that configuration resources covered by the lock are not modified by other NETCONF or non-NETCONF management operations such as SNMP and the CLI.

The duration of the partial lock is defined as beginning when the partial lock is granted. The partial lock lasts until either the corresponding <partial-unlock> operation succeeds or the NETCONF session terminates.

A NETCONF session MAY have multiple parts of one or more datastores (running, candidate,startup) locked using partial lock operations. The <partial-lock> operation returns a lock-id to identify each successfully acquired lock.

2.2. Dependencies

TOC

The device MUST support the restricted XPath expressions in the select element, as described in Section 2.4.1 (<partial-lock>). If optionally the :xpath capability is also supported, the device MUST also support the usage of any XPath 1.0 expression in the select element.

2.3. Capability Identifier

TOC

urn:ietf:params:netconf:capability:partial-lock:1.0

2.4. New Operations

TOC

2.4.1. <partial-lock>

TOC

The <partial-lock> operation allows the client to lock a portion of a data store. The portion to lock is specified by using XPath expressions in the select elements of the <partial-lock> operation. Each XPath expression MUST return a node set. Locking a node protects the complete subtree under it from modification by others.

The select XPath expressions are evaluated only once at lock time, thereafter the scope of the lock is maintained as a set of nodes. If the configuration data is later altered in a way that would make the original select XPath expressions evaluate to a different set of nodes, this does not affect the scope of the partial lock.

XPath is only used for the creation of the partial lock. Conceptually the scope of the lock is defined by the returned node set and not by the XPath expression.

A <partial-lock> operation MUST be handled atomically by the NETCONF server. The server either locks all requested parts of the data store or none; I.e. if during the operation one of the requested parts cannot be locked the server MUST unlock all parts that have been already locked during that operation.

If a node is locked by a session, only that same session is able to modify that node or any node in the subtree underneath it.

If a top level node of a locked subtree is deleted, any other session can recreate it, as it is not covered by the lock anymore. If all top level nodes are deleted, the lock will still be present, however it's scope will become nil i.e. it will not cover any nodes.

A partial lock operation MUST fail if:

- Any NETCONF session (including the current session) owns the global lock on the datastore.

* Any part of the scope to be locked is already locked by another management session/protocol, including other NETCONF sessions using the <partial-lock> or any other non-NETCONF management method.

* The NETCONF server implements access control and the locking user does not have sufficient privileges, to all parts of the datastore section to be locked. The exact handling of access rights is outside the scope of this document, but it is assumed that there is an access control system that MAY deny or allow the partial lock operation.

Note: If partial lock is requested for the running datastore, and the NETCONF server implements the :confirmed-commit capability, and there was a recent confirmed <commit> operation, where the confirming <commit> operation has not been received. In this case the lock MUST be denied, because if the confirmation does not arrive, the running datastore MUST be rolled back to its state before the commit, thus the NETCONF server might need to modify the configuration.

As with most locking systems, there is a possibility that two management sessions trying to lock different parts of the configuration become dead-locked. To avoid this situation, clients SHOULD lock everything they need in one operation. If that operation still fails, the client SHOULD back down, release any already acquired locks, and retry the procedure after some time interval. The length of the interval should preferably be random to avoid repeated dead-locks when both (or all) clients back down and then repeat locking.

It is the intention to keep partial-locking simple, so when a partial lock is executed you get what you asked for: a set of nodes that are locked for writing. There are some other issues that are intentionally not addressed for the sake of simplicity:

* Locking does not affect read operations.

* If a part of a datastore is locked, this has no effect on any unlocked parts of the datastore. If this is a problem e.g. the operator's changes depend on data values in the unlocked part of the datastore, the operator should include these values in the scope of the lock.

* An operator is allowed to edit the configuration both inside and outside the scope of a lock. It is the operator's responsibility to lock all parts of the datastore that are crucial for a specific management action.

Note: The <partial-lock> operation does not modify the global <lock>

operation defined in the base NETCONF Protocol [NETCONF] (Enns, R., “NETCONF Configuration Protocol,” December 2006.). If part of a datastore is already locked by <partial-lock>, then a global lock for that datastore fails even if the global lock is attempted by the same NETCONF session which owns the partial-lock.

2.4.1.1. Parameters, Result, Examples

Parameters:

> **target:** Name of the configuration datastore of which a part shall be locked. One <partial-lock> operation can affect only one of the datastores. URLs are not accepted.
>
> **select:** One or more 'select' elements each containing an XPath expression. The XPath expression is evaluated in a context where the context node is the root of the server's conceptual data model, and the set of namespace declarations are those in scope on the select element.
> The select expressions MUST each return a node set, and at least one of the node sets must be non-empty.
>
> If the device supports the :xpath capability as well any valid XPath 1.0 expression can be used, if not, the XPath expression MUST be limited to an Instance Identifier expression. An Instance Identifier is an absolute path expression in abbreviated syntax, where predicates are used only to specify values for nodes defined as keys to distinguish multiple instances.

Example: Lock virtual router 1 and interface eth1

```
         <nc:rpc
           xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0"
           xmlns="urn:ietf:params:xml:ns:netconf:partial-lock:1.0"
           xmlns:rte="http://example.com/ns/route">
           xmlns:if="http://example.com/ns/interface">
           nc:message-id="135">
             <partial-lock>
                 <nc:running/>
                 <select>
                     /routing/virtualRouter['routerName=router1']
                 </select>
                 <select>/interfaces/interface[Id='eth1']</select>
              </partial-lock>
         </nc:rpc>

         <nc:rpc-reply
           xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0"
           xmlns="urn:ietf:params:xml:ns:netconf:partial-lock:1.0"
           xmlns:rte="http://example.com/ns/route">
           xmlns:if="http://example.com/ns/interface">
           nc:message-id="135">
             <nc:data>
                 <lock-id>127</lock-id>
             </nc:data>
         </nc:rpc-reply>
```

Positive Response:

If the device was able to satisfy the request, an <rpc-reply> is sent with a <lock-id> element (lock identifier) in the <data> element.

Negative Response:

If a lock is already held on any node within the subtrees to be locked, the <error-tag> element is 'lock-denied' and the <error-info> element includes the <session-id> of the lock owner. If the lock is held by a non-NETCONF entity, a <session-id> of 0 (zero) is included.

If the select expressions return an empty node set, the <error-tag> is 'operation-failed', and the <error-app-tag> is 'no-matches'.

If any select expression returns anything but a node set, the <error-tag> is 'invalid-value', the <error-app-tag> is 'XPath does not return a node set'.

If the :xpath capability is not supported and the XPath expression is not an Instance Identifier, the <error-tag> is 'invalid-value', the <error-app-tag> is ':xpath capability not supported'.

If access control denies the partial lock, the <error-tag> is 'access-denied'.

2.4.1.2. Reserving model sections for future work

TOC

Partial lock can not be used to lock non-existing nodes, effectively reserving them for future use. To make sure that a node cannot be created by some other session, the parent node should be locked, the top level node of the new section created, and then locked with another <partial-lock> operation. After this the lock on the parent node should be removed.

2.4.2. <partial-unlock>

TOC

The operation unlocks a part of a datastore that was previously locked using <partial-lock> during the same session.

Parameters:

> **lock-id:** Lock identifier to unlock; taken from a reply to a previous <partial-lock> operation.

Example: Unlock

```
<nc:rpc xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0"
    xmlns="urn:ietf:params:xml:ns:netconf:partial-lock:1.0"
    nc:message-id="136">
    <partial-unlock>
        <lock-id>127</lock-id>
    </partial-unlock>
</nc:rpc>
```

Positive Response:

If the device was able to satisfy the request, an <rpc-reply> is sent that contains an <ok> element. A positive response MUST be sent even if all of the locked part of the datastore has already been deleted.

Negative Response:

If the <lock-id> parameter does not identify a lock which is owned by the session, an 'invalid-value' error is returned.

2.5. Modifications to Existing Operations

TOC

A granted partial-lock will cause another operation to fail, if it tries to modify the locked area and is executed in a NETCONF session other then the one that owns the lock. Affected operations include: <edit-config>, <copy-config>, <delete-config>, <commit> and <discard-changes>. A granted partial-lock will also cause the (global) <lock> operation to fail. All of these operations are affected only if they are for the same datastore.

2.6. Interactions with Other Capabilities

TOC

2.6.1. Writable-Running Capability

TOC

Partial locking of the running datastore can only be done if the :writable-running capability is supported by the device.

2.6.2. Candidate Configuration Capability

TOC

Partial locking of the candidate datastore can only be done if the :candidate capability is supported by the device. The partial locking of the candidate datastore does not depend on whether the datastore was modified or not.

2.6.3. Distinct Startup Capability

TOC

Partial locking of the startup datastore can only be done if the :startup capability is supported by the device.

TOC

3. Security Considerations

The same considerations as for the base NETCONF Protocol [NETCONF] (Enns, R., “NETCONF Configuration Protocol,” December 2006.) are valid. It is assumed that the <partial-lock> and <partial-unlock> RPCs are only allowed for an authenticated user after passing some access control mechanism.

A lock either a partial-lock or a global lock might prevent other users from configuring the system. The following mechanisms are in place to prevent the misuse of this possibility:

> Only an authenticated user after passing access control can request a partial-lock.
>
> The partial-lock is automatically released when a session is terminated irrespective of the manner the session ends.
>
> The <kill-session> operation allows terminating other users sessions.
>
> The NETCONF server may log partial-lock requests in an audit trail.

Partial locking is NOT an authorization mechanism, it should not be used to provide security or access control. Partial locking should only be used as a mechanism to provide consistency in case of multiple managers trying to configure the node. It is vital that the operator can easily understand the exact scope of a lock, for this reason the scope is determined when granting a lock and is not modified dynamically later.

---

4. IANA Considerations

This document registers two URIs for the NETCONF XML namespace in the IETF XML registry [RFC3688] (Mealling, M., “The IETF XML Registry,” January 2004.). Note that the capability URN is compliant to [NETCONF] (Enns, R., “NETCONF Configuration Protocol,” December 2006.) section 10.3.

Index	Capability Identifier
:partial-lock	urn:ietf:params:netconf:capability:partial-lock:1.0

URI: urn:ietf:params:xml:ns:netconf:partial-lock:1.0

Registrant Contact: The IESG.

XML: N/A, the requested URI is an XML namespace.

5. Appendix A - XML Schema for Partial Locking (normative)

TOC

The following XML Schema defines the <partial-lock> and <partial-unlock> operations:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns="urn:ietf:params:xml:ns:netconf:partial-lock:1.0"
  xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0"
  targetNamespace="urn:ietf:params:xml:ns:netconf:partial-lock:1.0"
  elementFormDefault="qualified" attributeFormDefault="unqualified">

  <xs:annotation>
    <xs:documentation>
        Schema defining the partial-lock and unlock operations.
        organization "IETF NETCONF Working Group"
        contact
            "Balazs Lengyel
             Ericsson Hungary, Inc.
             balazs.lengyel@ericsson.com"
     </xs:documentation>
  </xs:annotation>

  <xs:import namespace="urn:ietf:params:xml:ns:netconf:base:1.0"
    schemaLocation="netconf.xsd"/>

  <xs:complexType name="partialLockType">
    <xs:annotation>
      <xs:documentation>
        A NETCONF operation that locks part of a datastore.
      </xs:documentation>
    </xs:annotation>
    <xs:complexContent>
      <xs:extension base="nc:rpcOperationType">
        <xs:sequence>
          <xs:element ref="nc:config-name"/>
          <xs:element name="select" type="xs:string"
            maxOccurs="unbounded">
            <xs:annotation>
              <xs:documentation>
                XPath expression that specifies the scope of the lock.
                An Instance Identifier expression must be used unless
                the :xpath capability is supported in which case any
                XPath 1.0 expression is allowed.
              </xs:documentation>
            </xs:annotation>
          </xs:element>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>

  <xs:complexType name="partialUnLockType">
```

```
        <xs:annotation>
          <xs:documentation>
            A NETCONF operation that releases a previously acquired
            partial-lock.
          </xs:documentation>
        </xs:annotation>
        <xs:complexContent>
          <xs:extension base="nc:rpcOperationType">
            <xs:sequence>
              <xs:element name="lock-id" type="xs:unsignedInt">
                <xs:annotation>
                  <xs:documentation>
                    Identifies the lock to be released. Must be the value
                    received in the response to the partial-lock operation.
                  </xs:documentation>
                </xs:annotation>
              </xs:element>
            </xs:sequence>
          </xs:extension>
        </xs:complexContent>
      </xs:complexType>

      <!-- <partial-lock> operation -->
      <xs:element name="partial-lock" type="partialLockType"
                  substitutionGroup="nc:rpcOperation"/>

      <!-- <partial-unlock> operation -->
      <xs:element name="partial-unlock" type="partialUnLockType"
                  substitutionGroup="nc:rpcOperation"/>

      <!-- reply to <partial-lock> -->
      <xs:element name="lock-id" type="xs:unsignedInt"/>

    </xs:schema>
```

6. Appendix B - YANG Module for Partial Locking (non-normative)

TOC

The following YANG module defines the <partial-lock> and <partial-unlock> operations. The YANG language is defined in [I-D.ietf-netmod-yang] (Bjorklund, M., “YANG - A data modeling language for NETCONF,” May 2008.).

```
module netconf-partial-lock {

  namespace urn:ietf:params:xml:ns:netconf:partial-lock:1.0;
  prefix pl;

  organization "IETF NETCONF Working Group";

  contact
    "Balazs Lengyel
     Ericsson Hungary, Inc.
     balazs.lengyel@ericsson.com";

  description
    "This YANG module defines the <partial-lock> and
     <partial-unlock> operations.";

  revision 2008-06-06 {
    description "Initial version.";
  }

  grouping configName {
    description
      "A choice to list the datastore names for NETCONF.
       This could be moved to a netconf.yang module.";
    choice configNameType {
      leaf running   { type empty; }
      leaf candidate { type empty; }
      leaf startup   { type empty; }
    }
  }

  rpc partial-lock {
    description "A NETCONF operation that locks part of a datastore.";
    input {
      uses configName;
      leaf-list select {
        description
          "XPath expression that specifies the scope of the lock.
          An Instance Identifier expression must be used unless the
          :xpath capability is supported in which case any XPath 1.0
          expression is allowed.";
        type string;
        min-elements 1;
      }
    }
    output {
      leaf lock-id {
        description
          "Identifies the lock, if granted. The lock-id must be
```

```
              used in the partial-unlock rpc.";
           type uint32;
         }
       }
     }

     rpc partial-unlock {
       description
         "A NETCONF operation that releases a previously acquired
         partial-lock.";
       input {
         leaf lock-id {
           description
             "Identifies the lock to be released. Must be the value
             received in the response to the partial-lock operation.";
           type uint32;
         }
       }
     }
   }
```

7. Appendix C - Change Log

TOC

7.1. 02-03

TOC

Minor clarifications
Same descriptions in XSD and YANG.

7.2. 01-02

TOC

Made XSD normative
Clarified that no specific access control is assumed.
Clarified that non-existing nodes are NOT covered by the lock, even if they where existing and covered by the lock when it was originally granted.
Some rewording
Added app-tags for two of the error cases.
Made YANG an informative reference
Enhanced security considerations.

7.3. 00-01

TOC

Added YANG module.

7.4. -00

TOC

Created from draft-lengyel-ngo-partial-lock-01.txt

8. Acknowledgements

TOC

Thanks to Andy Bierman, Sharon Chisholm, Phil Shafer , David Harrington, Mehmet Ersue and many other members of the NETCONF WG for providing important input to this document.

9. References

TOC

9.1. Normative References

TOC

[NETCONF]	Enns, R., “NETCONF Configuration Protocol,” RFC 4741, December 2006 (TXT).
[RFC2119]	Bradner, S., “Key words for use in RFCs to Indicate Requirement Levels,” BCP 14, RFC 2119, March 1997 (TXT, HTML, XML).
[RFC3688]	Mealling, M., “The IETF XML Registry,” BCP 81, RFC 3688, January 2004 (TXT).

9.2. Informative References

TOC

[I-D.ietf-netmod-yang]	Bjorklund, M., “YANG - A data modeling language for NETCONF,” draft-ietf-netmod-yang-00 (work in progress), May 2008 (TXT).

Authors' Addresses

TOC

	Balazs Lengyel
	Ericsson Hungary
Email:	balazs.lengyel@ericsson.com
	Martin Bjorklund
	Tail-f Systems
Email:	mbj@tail-f.com

Full Copyright Statement

TOC

Intellectual Property