NETCONF
Internet-Draft
Intended status: Standards Track
Expires: January 3, 2010

B. Lengyel
Ericsson
M. Bjorklund
Tail-f Systems
July 02, 2009

Partial Lock RPC for NETCONF
draft-ietf-netconf-partial-lock-09

Status of this Memo

Copyright Notice

Abstract

The NETCONF protocol defines the lock and unlock RPCs, used to 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

The [NETCONF] protocol describes the lock and unlock operations 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 a capability based extension to the NETCONF protocol to support partial locking of NETCONF datastores using a mechanism based on the existing XPath filtering mechanisms.

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].

Additionally the following terms are defined:

- Instance Identifier: an XPath expression identifying a specific node in the conceptual XML datastore. It contains an absolute path expression in abbreviated syntax, where predicates are used only to specify values for nodes defined as keys to distinguish multiple instances.

- Scope of the lock: initially the set of nodes returned by the XPath expressions in a successful partial-lock operation. The set might be modified if some of the nodes are deleted.

- Protected area: the set of nodes that are protected from modification by the lock. This set consists of nodes in the scope of the lock and nodes in subtrees under them.

2. Partial Locking Capability

2.1. Overview

The :partial-lock capability indicates that the device supports the locking of its configuration with a more limited scope than a complete configuration datastore. The scope to be locked is specified by using restricted or full XPath expressions. Partial locking only affects configuration 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 begins when the partial lock is granted and lasts until (1) either the corresponding <partial-unlock> operation succeeds or (2) 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. The lock-id is unique for a NETCONF server for all partial-locks granted to any NETCONF or non-NETCONF sessions.

2.1.1. Usage Scenarios

In the following we describe a few scenarios for partial locking. Partial locking is primarily useful towards the running configuration. However it can be used to lock parts of a candidate datastore as well. While scenarios using the running datastore are seen as the most important, as an example a scenario involving the candidate datastore is also presented. Besides the three described here, there are many other usage scenarios possible.

2.1.1.1. Multiple managers handling the writable running datastore with overlapping sections

Multiple managers are handling the same NETCONF agent simultaneously. The agent is handled via the writable running datastore. Each manager has his or her own task, which might involve the modification of overlapping sections of the datastore.

After collecting and analyzing input and preparing the NETCONF operations off-line, the manager locks the areas that are important for his task using one single <partial-lock> operation. The manager executes a number of <edit-config> operations to modify the configuration, then releases the partial-lock. The lock should be held for the shortest possible time (e.g. seconds rather then minutes). The manager should collect all human input before locking anything. As each manager locks only a part of the data model, usually multiple operators can execute the <edit-config> operations simultaneously.

2.1.1.2. Multiple managers handling the writable running datastore, distinct management areas

Multiple managers are handling the same NETCONF agent simultaneously. The agent is handled via the writable running datastore. The agent's data model contains a number of well defined separate areas that can be configured without impacting other areas. An example can be a

server with multiple applications running on it, or a number of a network elements with a common NETCONF agent for management.

Each manager has his or her own task, which does not involve the modification of overlapping sections of the datastore.

The manager locks his area with a <partial-lock> operation, uses a number of <edit-config> commands to modify it, later releases the lock. As each manager has his functional area assigned to him, and he locks only that area, multiple managers can edit the configuration simultaneously. Locks can be held for extended periods (e.g. minutes, hours), as this will not hinder other managers.

This scenario assumes, that the global lock operation from [NETCONF] is not used.

2.1.1.3. Multiple managers handling the candidate datastore in a semi-coordinated work mode

Multiple managers are handling the same NETCONF agent simultaneously. The agent is handled via the candidate datastore. Each manager has his or her own task which might involve the modification of overlapping sections of the datastore.

After collecting and analyzing input and preparing the NETCONF operations off-line, the manager locks the areas that are important for his task using one single <partial-lock> operation in both the candidate and the running datastore. He executes a number of <edit-config> operations to modify the configuration, then releases the partial-lock. The lock should only be held for the shortest possible time (e.g. seconds rather then minutes).

Operators coordinate with each other. When all of them finish their tasks one of them orders commit. If any of the operators are still working, and holds a lock, the commit will fail, and will need to be repeated after all managers finish.

Warning: When multiple managers use the candidate configuration in parallel, there is a risk that the interaction of access control (which is still implementation specific at the time of this writing) and the commit operation might result in a manager becoming unable both to commit or discard changes, as illustrated by the following sequence.

Manager A only has access to the interfaces branch in the model, and edits it in candidate

Manager B only has access to the routing branch in the model, and edits it in candidate

Manager A terminates it's session

Now Manager B can not issue <commit> because it can not modify interfaces in the running datastore

Manager B can not issue <discard-changes> because it can not modify interfaces in the candidate datastore

The situation is not a result of partial locking as a lock can be easily removed; it is the result of a potential interaction between access control, which by nature is specific for different parts of the datastore and the global nature of the commit operation.

2.2. Dependencies

The device MUST support restricted XPath expressions in the select element, as described in Section 2.4.1. Optionally, if the :xpath capability is also supported (as defined in [NETCONF] chapter 8.9. XPath Capability), the device MUST also support using any XPath 1.0 expression in the select element.

2.3. Capability Identifier

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

2.4. New Operations

2.4.1. <partial-lock>

The <partial-lock> operation allows the client to lock a portion of one or more datastores. The portion to lock is specified with XPath expressions in the "select" elements and the list of datastores in the "target" elements in the <partial-lock> operation. Each XPath expression MUST return a node set.

When a NETCONF session holds a lock on a node, no other session or non-NETCONF mechanism of the system can change that node or any node in the hierarchy of nodes beneath it.

Locking a node protects the node itself and the complete subtree under the node from modification by others. The set of locked nodes is called the scope of the lock, while all the locked nodes and the nodes in the subtrees under them make up the protected area.

In some situations it is desirable that the same set of nodes are

locked in more than one datastore. For example, if an interface is configured in the candidate datastore, it is dangerous for it to be configured by someone else in a possibly conflicting manner in the running datastore. For this reason <partial-lock> allows the locking of the same sections of the management data in multiple datastores.

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

Let's say the agent's data model includes a list of interface nodes. If the XPath expression in the partial lock operation covers all interface nodes at locking, the scope of the lock will be maintained as the list of interface nodes at the time when the lock was granted. If someone later creates a new interface, this new interface will not be included in the locked-nodes list created previously, the new interface will not be locked.

A <partial-lock> operation MUST be handled atomically by the NETCONF server. The server either locks all requested parts of the datastore(s) or none. If during the <partial-lock> operation one of the requested parts cannot be locked, the server MUST unlock all parts that have already been locked during that operation.

If a node in the scope of the lock is deleted, it is removed from the scope of the lock, so any other session or non-NETCONF mechanism can recreate it. If all nodes in the scope of the lock are deleted, the lock will still be present. However, its scope will become empty (since the lock will not cover any nodes).

A NETCONF server MUST be able to grant multiple simultaneous partial locks to a single NETCONF session. If the protected area of the individual locks overlaps, nodes in the common area MUST be protected until all of the locks are released.

A partial lock operation MUST fail if:

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

- Any part of the area to be protected is already locked (or protected by partial locking) by another management session, including other NETCONF sessions using <partial-lock> or any other non-NETCONF management method.

- The NETCONF server implements access control, and the locking user does not have sufficient access rights. 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.

The <partial-lock> operation is designed for simplicity, so when a partial lock is executed you get what you asked for: a set of nodes that are locked for writing. As a consequence users must observe the following:

- Locking does not affect read operations.

- If part of a datastore is locked, this has no effect on any unlocked parts of the datastore. If this is a problem (e.g., changes depend on data values or nodes outside the protected part of the datastore), these nodes SHOULD be included in the protected area of the lock.

- Configuration data can be edited both inside and outside the protected area of a lock. It is the responsibility of the NETCONF client application to lock all relevant parts of a 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]. If part of a datastore is already locked by <partial-lock>, then a global lock for that datastore MUST fail even if the global lock is requested by the NETCONF session that owns the partial lock.

2.4.1.1. Parameters, Result, Examples

Parameters:

target: Name of one or more configuration datastores of which a part shall be locked. If multiple datastores are specified the same select parameter(s) are evaluated for each of them.

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.

Each select expression is evaluated for each targeted datastore.

The nodes returned from the select expressions are reported in

the rpc-reply message.

Note that if some of the requested nodes exist only in some of the targeted datastores, the lock is granted on different nodes in different datastores.

Each select expression MUST return a node set, and at least one of the node sets for one of the specified datastores MUST be non-empty.

If the device supports the :xpath capability, any valid XPath 1.0 expression can be used. If the device does not support the :xpath capability, 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="urn:ietf:params:xml:ns:netconf:partial-lock:1.0"
   xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0"
   message-id="135">
     <partial-lock>
         <target>
             <running/>
         </target>
         <select xmlns:rte="http://example.com/ns/route">
             /rte:routing/rte:virtualRouter[rte:routerName='router1']
         </select>
         <select xmlns:if="http://example.com/ns/interface">
             /if:interfaces/if:interface[if: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"
   message-id="135">
     <lock-id>127</lock-id>
     <running>
       <locked-node xmlns:rte="http://example.com/ns/route">
           /rte:routing/rte:virtualRouter[rte:routerName='router1']
       </locked-node>
       <locked-node xmlns:if="http://example.com/ns/interface">
           /if:interfaces/if:interface[if:id='eth1']
       </locked-node>
```

```
     </running>
  </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 <rpc-reply> element. A list of locked nodes per datastore is also returned in Instance Identifier format.

Negative Response:

If a lock is already held by another session 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 session, a <session-id> of 0 (zero) is included. If needed the returned session-id may be used to <kill-session> the NETCONF session holding the lock. The same error response is returned if the requesting session already holds the (global) lock for the same datastore.

If any select expression is an invalid XPath expression, the <error-tag> is 'invalid-value'.

If any select expression returns something other than a node set, the <error-tag> is 'invalid-value', and the <error-app-tag> is 'not-a-node-set'.

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

If any of the target datastores does not exist, the <error-tag> is 'invalid-value', the <error-app-tag> is 'invalid-lock-specification'

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 'invalid-lock-specification'.

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

2.4.1.2. Deadlock Avoidance

As with most locking systems, it is possible that two management sessions trying to lock different parts of the configuration could become dead-locked. To avoid this situation, clients SHOULD lock everything they need in one operation. If locking fails, the client MUST back-off, release any previously acquired locks, and SHOULD retry the procedure after waiting some randomized time interval.

2.4.2. <partial-unlock>

The operation unlocks the parts of the datastores that were previously locked using <partial-lock> during the same session.

Parameters:

lock-id: Identity of the lock to be unlocked. This lock-id MUST have been received as a response to a lock request by the manager during the current session, and MUST NOT have been sent in a previous unlock request.

Example: Unlock a previously created lock

```
<nc:rpc xmlns="urn:ietf:params:xml:ns:netconf:partial-lock:1.0"
  xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0"
  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 parts of the datastore(s) have 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

A successful partial lock will cause a subsequent operation to fail if that attempts to modify nodes in the protected area of the lock and is executed in a NETCONF session other than the session that has been granted the lock. The <error-tag> 'in-use' and the <error-app-tag> 'locked' is returned. All operations that modify the datastore are affected, including: <edit-config>, <copy-config>, <delete-config>, <commit> and <discard-changes>. If partial lock prevents <edit-config> modifying some data, but the operation includes the continue-on-error option, modification of other parts of the datastore, which are not protected by partial locking, might still succeed.

If a datastore contains nodes locked by partial lock, this will cause the (global) <lock> operation to fail. The <error-tag> element 'lock-denied' and an <error-info> element including the <session-id> of the lock owner will be returned. If the lock is held by a non-NETCONF session, a <session-id> of 0 (zero) is returned.

All of these operations are affected only if they are targeting the same datastore.

2.6. Interactions with Other Capabilities

2.6.1. Candidate Configuration Capability

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

2.6.2. Confirmed Commit Capability

If:

- a 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

then 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. The NETCONF server might therefore need to modify the configuration.
In this case the <error-tag> 'in-use' and the <error-app-tag> 'outstanding-confirmed-commit' is returned.

2.6.3. Distinct Startup Capability

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

3. Security Considerations

The same considerations are relevant as for the base NETCONF Protocol [NETCONF]. 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 and authorized user can request a partial lock.

The partial lock is automatically released when a session is terminated regardless of how the session ends.

The <kill-session> operation makes it possible to terminate other

users's sessions.

The NETCONF server may log partial lock requests in an audit trail.

A lock that is hung for some reason (e.g., a broken TCP connection that the server has not yet recognised) can be released using another NETCONF session by explicitly killing the session owning that lock using the <kill-session> operation.

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 for providing consistency when multiple managers are trying to configure the node. It is vital that users easily understand the exact scope of a lock. This is why the scope is determined when granting a lock and is not modified thereafter.

4. IANA Considerations

This document registers one capability identifier URN from the "Network Configuration Protocol (NETCONF) Capability URNs" registry, and one URI for the NETCONF XML namespace in the "IETF XML registry" [RFC3688]. Note that the capability URN is compliant to [NETCONF] 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)

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
        Netconf Working Group
        Mailing list: netconf@ietf.org
        Web: http://www.ietf.org/html.charters/netconf-charter.html

        Balazs Lengyel
        balazs.lengyel@ericsson.com"

      revision 2009-02-19
        description Initial version, published as RFC yyyy.
-- RFC Ed.: replace yyyy with actual RFC number and remove this note.
    </xs:documentation>
  </xs:annotation>

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

  <xs:simpleType name="lock-id-type">
    <xs:annotation>
      <xs:documentation>
        A number identifying a specific
        partial-lock granted to a session.
        It is allocated by the system, and SHOULD
        be used in the unlock operation.
      </xs:documentation>
    </xs:annotation>
    <xs:restriction base="xs:unsignedInt"/>
  </xs:simpleType>

  <xs:complexType name="configNameType"/>
```

```
   <xs:element name="config-name"
```



```
       type="configNameType" abstract="true"/>
   <xs:element name="startup" type="configNameType"
       substitutionGroup="config-name"/>
   <xs:element name="candidate" type="configNameType"
       substitutionGroup="config-name"/>
   <xs:element name="running" type="configNameType"
       substitutionGroup="config-name"/>

   <xs:complexType name="partialLockType">
     <xs:annotation>
       <xs:documentation>
         A NETCONF operation that locks part of one or more datastores.
       </xs:documentation>
     </xs:annotation>
     <xs:complexContent>
       <xs:extension base="nc:rpcOperationType">
         <xs:sequence>
           <xs:element name="target" maxOccurs="unbounded">
             <xs:annotation>
               <xs:documentation>
                 A list of one or more datastore names for NETCONF.
                 Each target element MUST contain a different
                 datastore name.
               </xs:documentation>
             </xs:annotation>
             <xs:complexType>
               <xs:sequence>
                   <xs:element ref="config-name"/>
               </xs:sequence>
             </xs:complexType>
           </xs:element>
           <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="lock-id-type">
            <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:complexType name="contentPartInPartialLockReplyType">
```

```
      <xs:annotation>
        <xs:documentation>
          The content of the reply to a successful
          partial-lock request MUST conform to this complex type.
        </xs:documentation>
      </xs:annotation>
      <xs:sequence>
        <xs:element name="lock-id" type="lock-id-type">
          <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:element name="running" minOccurs="0">
          <xs:annotation>
            <xs:documentation>
              List of locked nodes in the running datastore.
            </xs:documentation>
          </xs:annotation>
          <xs:complexType>
            <xs:sequence>
              <xs:element name="locked-node" type="xs:string"
                maxOccurs="unbounded">
              </xs:element>
            </xs:sequence>
          </xs:complexType>
        </xs:element>
        <xs:element name="candidate" minOccurs="0">
          <xs:annotation>
            <xs:documentation>
              List of locked nodes in the candidate datastore.
            </xs:documentation>
          </xs:annotation>
          <xs:complexType>
            <xs:sequence>
              <xs:element name="locked-node" type="xs:string"
                maxOccurs="unbounded">
              </xs:element>
            </xs:sequence>
```

```
        </xs:complexType>
      </xs:element>
      <xs:element name="startup" minOccurs="0">
        <xs:annotation>
          <xs:documentation>
            List of locked nodes in the startup datastore.
          </xs:documentation>
        </xs:annotation>
        <xs:complexType>
          <xs:sequence>
            <xs:element name="locked-node" type="xs:string"
              maxOccurs="unbounded">
            </xs:element>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>

 </xs:schema>
```


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

The following YANG module defines the <partial-lock> and <partial-unlock> operations. The YANG language is defined in [I-D.ietf-netmod-yang].

```
module ietf-netconf-partial-lock {

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

  organization "IETF Network Configuration (netconf) Working Group";

  contact
   "Netconf Working Group
    Mailing list: netconf@ietf.org
    Web: http://www.ietf.org/html.charters/netconf-charter.html

    Balazs Lengyel
```

```
    Ericsson
    balazs.lengyel@ericsson.com";

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

  revision 2009-02-19 {
    description
     "Initial version, published as RFC yyyy.";
     // RFC Ed.: replace yyyy with actual RFC number & remove this note.
  }

  typedef lock-id-type {
    type uint32;
    description
     "A number identifying a specific partial-lock granted to a session.
      It is allocated by the system, and SHOULD be used in the
      partial-unlock operation.";
  }

  rpc partial-lock {
    description
     "A NETCONF operation that locks part of one or more datastores.";
    input {
      list target {
        min-elements 1;
        description
         "A list of one or more datastore names.
```



```
          Each list entry must contain a different datastore name.";
        choice datastore {
          leaf running   { type empty; }
          leaf candidate { type empty; }
          leaf startup   { type empty; }
        }
      }
      leaf-list select {
        type string;
        min-elements 1;
        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.";
      }
    }
    output {
      leaf lock-id {
        type lock-id-type;
        description
         "Identifies the lock, if granted. The lock-id SHOULD be
          used in the partial-unlock rpc.";
      }
      container running {
        leaf-list locked-node {
          type instance-identifier;
          min-elements 1;
          description
           "List of locked nodes in the running datastore";
        }
      }
      container candidate {
        leaf-list locked-node {
          type instance-identifier;
          min-elements 1;
          description
           "List of locked nodes in the candidate datastore";
        }
      }
      container startup {
        leaf-list locked-node {
          type instance-identifier;
          min-elements 1;
          description
           "List of locked nodes in the startup datastore";
        }
      }
```



```
    }
  }

  rpc partial-unlock {
    description
```

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

7. Appendix C - Usage Example - Reserving nodes for future editing (non-normative)

Partial lock cannot be used to lock non-existent nodes, which would effectively attempt to reserve them for future use. To guarantee that a node cannot be created by some other session, the parent node should be locked, the top level node of the new subtree created, and then locked with another <partial-lock> operation. After this, the lock on the parent node should be removed.

In this chapter an example illustrating the above is given.

We want to create <user> Joe under <users>, and start editing it. Editing might take a number of minutes. We want to immediately lock Joe so no one will touch it before we are finished with the editing.

We also want to minimize locking other parts of the datastore as multiple managers might be adding users near simultaneously.

First we check what users are already defined.

Step 1 - Read existing users

```
<rpc message-id="101"
    xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <get-config>
    <source>
      <running/>
    </source>
    <filter type="subtree">
      <top xmlns="http://example.com/users">
        <users/>
      </top>
    </filter>
  </get-config>
</rpc>
```

The NETCONF server sends the following reply.

Step 2 - Receiving existing data

```
<rpc-reply message-id="101"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <data>
    <top xmlns="http://example.com/users">
      <users>
        <user>
          <name>fred</name>
          <phone>8327</phone>
        </user>
      </users>
    </top>
  </data>
</rpc-reply>
```

We want to add the new user "Joe" and immediately lock him using partial locking. The way to do this, is to first lock all <user> nodes by locking the <users> node.

Note that if we would lock all the <user> nodes using the select expression '/usr:top/usr:users/usr:user' ; this would not lock the new user "Joe", which we will create after locking. So we rather have to lock the <users> node.

Step 3 - Lock users

```
<nc:rpc
  xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0"
  xmlns="urn:ietf:params:xml:ns:netconf:partial-lock:1.0"
      message-id="102">
  <partial-lock>
    <target>
      <running/>
    </target>
    <select xmlns:usr="http://example.com/users">
      /usr:top/usr:users
    </select>
  </partial-lock>
```

```
</nc:rpc>
```

The NETCONF server grants the partial lock. The scope of the lock includes only the <users> node. The lock protects the <users> node and all <user> nodes below it from modification (by other sessions).

Step 4 - Receive lock

```
<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"
  message-id="102">
    <lock-id>1</lock-id>
    <running>
      <locked-node xmlns:usr="http://example.com/users">
          /usr:top/usr:users
      </locked-node>
    </running>
</nc:rpc-reply>
```

Next we create user Joe. Joe is protected by the lock received above, as it is under the sub-tree rooted at the <users> node.

Step 5 - Create user Joe

```
<rpc message-id="103"
     xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <edit-config>
    <target>
      <running/>
    </target>
    <config>
      <top xmlns:usr="http://example.com/users">
        <users>
          <user>
            <name>Joe</name>
          </user>
        </users>
      </top>
    </config>
```

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

We receive a positive reply to the <edit-config> (not shown). Next we request a lock, that locks only <user> Joe, and release the lock on the <users> node. This will allow other managers to create additional new users.

Step 6 - Lock user Joe

```
<nc:rpc
  xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0"
  xmlns="urn:ietf:params:xml:ns:netconf:partial-lock:1.0"
      message-id="104">
  <partial-lock>
    <target>
      <running/>
    </target>
    <select xmlns:usr="http://example.com/users">
      /usr:top/usr:users/user[usr:name="Joe"]"
    </select>
  </partial-lock>
</nc:rpc>
```

The NETCONF server grants the partial lock. The scope of this second lock includes only the <user> node with name Joe. The lock protects all data below this particular <user> node.

Step 7 - Receive lock

```
<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"
  message-id="104">
    <lock-id>2</lock-id>
```

```
    <running>
      <locked-node xmlns:usr="http://example.com/users">
          /usr:top/usr:users/user[usr:name="Joe"]"
      </locked-node>
    </running>
</nc:rpc-reply>
```

The scope of the second lock is the <user> node Joe. It protects this <user> node and any data below it (e.g. phone number). At this point of time these nodes are protected both by the first and second lock. Next we unlock the other <user>s and the <users> node, to allow other managers to work on them. We still keep the second lock, so the <user> node Joe and the sub-tree below is still protected.

Step 8 - Release lock on <users>

```
<nc:rpc xmlns="urn:ietf:params:xml:ns:netconf:partial-lock:1.0"
    xmlns:nc="urn:ietf:params:xml:ns:netconf:base:1.0"
    message-id="105">
  <partial-unlock>
    <lock-id>1</lock-id>
  </partial-unlock>
</nc:rpc>
```

8. Appendix D - Change Log

8.1. 08-09

Clarifications

8.2. 07-08

Clarifications

8.3. 06-07

Changed XSD and YANG to allow additional proprietary datastores to be locked.

8.4. 05-06

Added usage example

Clarified error messages

Clarified interaction with edit-config continue-on-error

Improved YANG: indentation, canonical order, contact info

Added usage example in appendix C

Synchronized YANG and XSD

8.5. 04-05

Language and editorial updates

all app-tags are with dashes without spaces

Added usage scenarios

Changed encoding

Clarified definitions, separated scope of lock and protected area

8.6. 03-04

Minor clarifications

Added list of locked-nodes to the output of partial-lock.

Added <target> wrapper around datastore names.

Allowed atomic/one operation locking of datastore parts in multiple datastores.

Improved English (hopefully)

Removed the <data> element from rpc-reply following the text of rfc4741.

8.7. 02-03

Minor clarifications

Same descriptions in XSD and YANG.

8.8. 01-02

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.

8.9. 00-01

Added YANG module.

8.10. -00

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

9. Acknowledgements

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

10. References

10.1. Normative References

[NETCONF] Enns, R., "NETCONF Configuration Protocol", RFC 4741, December 2006.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, January 2004.

10.2. Informative References

[I-D.ietf-netmod-yang] Bjorklund, M., "YANG - A data modeling language for NETCONF", draft-ietf-netmod-yang-06 (work in progress), June 2009.

Authors' Addresses

Balazs Lengyel
Ericsson

Email: balazs.lengyel@ericsson.com

Martin Bjorklund
Tail-f Systems

Email: mbj@tail-f.com